intel APPLICATION NOTES AP-90

MASTER

May 1980

An 8741A/8041A Digital Cassette Controller

John Beaston, Jim Kahn
Peripheral Applications

An 8741A/8041A Digital Cassette Controller

Contents

INTRODUCTION

The microcomputer system designer requiring a low-cost, non-volatile storage medium has a difficult choice. His options have been either relatively expensive, as with floppy discs and bubble memories, or non-transportable, like battery backed-up RAMs. The full-sized digital cassette option was open but many times it too was too expensive for the application. Filling this void of low-cost storage is the recently developed digital mini-cassette. These mini-cassettes are similar to, but not compatible with, dictation cassettes. The mini-cassette transports are inexpensive (well under $100 in quantity), small (less than 25 cu. in.), low-power (one watt), and their storage capacity is a respectable 200K bytes of unformatted data on a 100-foot tape. These characteristics make the mini-cassette perfect for applications ranging from remote datalogging to program storage for hobbyist systems.

The only problem associated with mini-cassette drives is controlling them. While these drives are relatively easy to interface to a microcomputer system, via a parallel I/O port, they can quickly overburden a CPU if other concurrent or critical real-time I/O is required. The cleanest and probably the least expensive solution in terms of development cost is to use a dedicated single-chip controller. However, a quick search through the literature turns up no controllers compatible with these new transports. What to do? Enter the UPI-41A family of Universal Peripheral Interfaces.

The UPI-41A family is a group of two user-programmable slave microcomputers plus a companion I/O expander. The 8741A is the "flag-chip" of the line. It is a complete microcomputer with 1024 bytes of EPROM program memory, 64 bytes of RAM data memory, 16 individually programmable I/O lines, an 8-bit event counter and timer, and a complete slave peripheral interface with two interrupts and Direct Memory Access (DMA) control. The 8041A is the masked ROM, pin compatible version of the 8741A. Figure 2 shows a block diagram common to both parts. The 8243 I/O port expander completes the family. Each 8243 provides 16 programmable I/O lines.

Using the UPI concept, the designer can develop a custom peripheral control processor for his particular I/O problem. The designer simply develops his peripheral control algorithm using the UPI-41A assembly language and programs the EPROM of

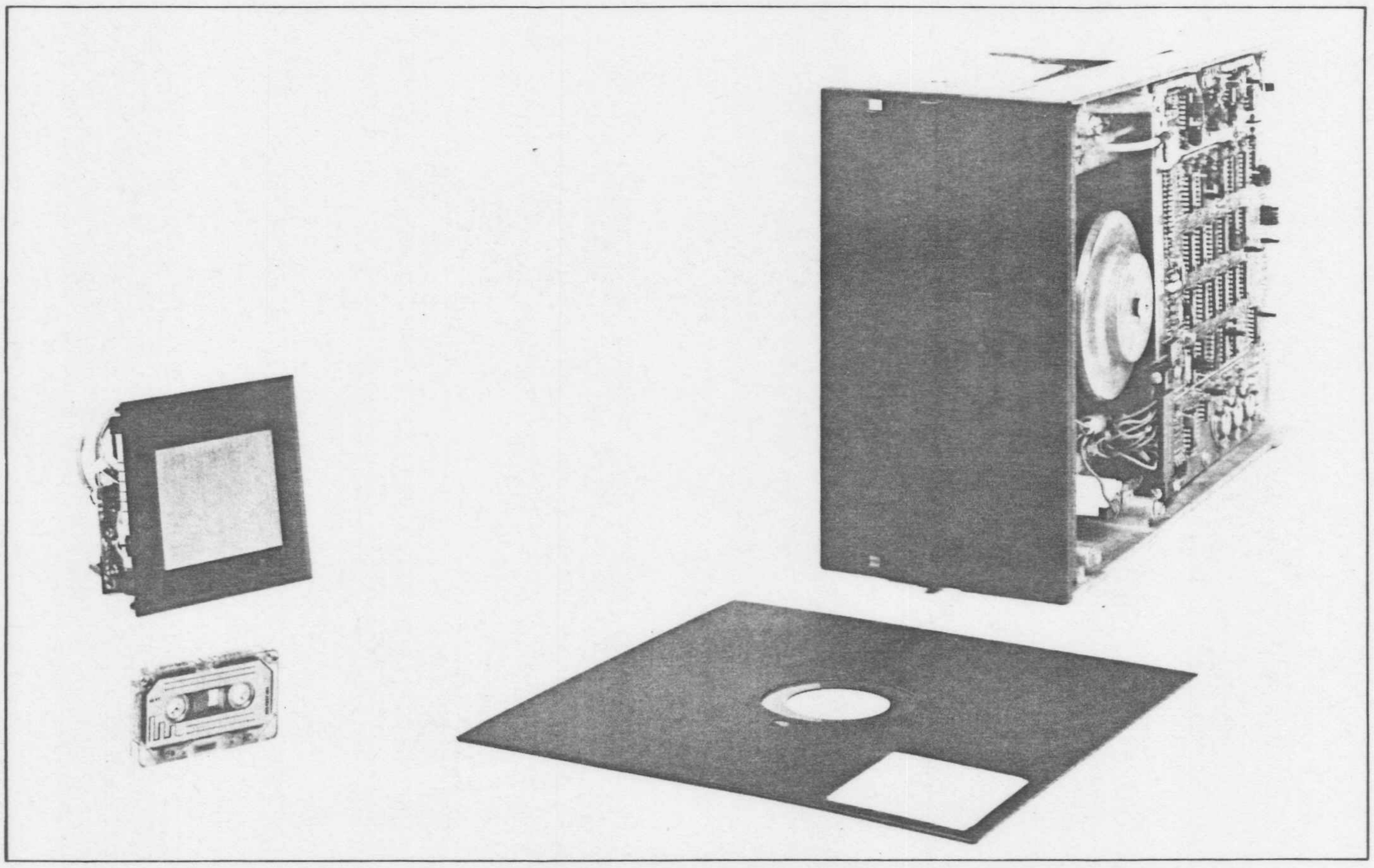
Figure 1. Comparison of Mini-Cassette and Floppy Disk Transports and Media.

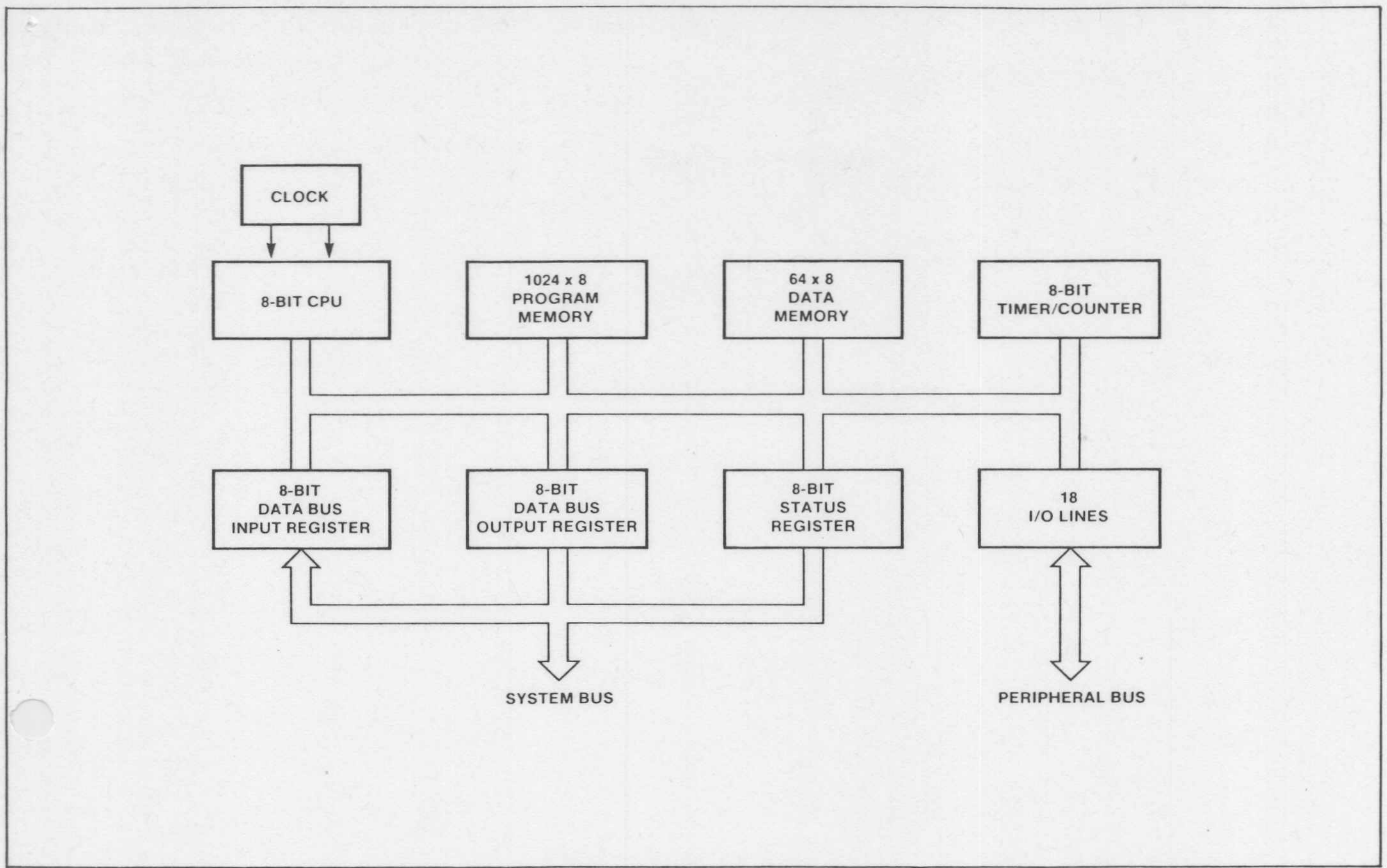

Figure 2. 8741A/8041A Block Diagram

the 8741A. Voila! He has a single-chip dedicated controller. Testing may be accomplished using either an ICE-41A or the Single-step mode of the 8741A. UPI-41A peripheral interfaces are being used to control printers, keyboards, displays, custom serial interfaces, and data encryption units. Of course, the UPI family is perfect for developing a dedicated controller for digital mini-cassette transports. To illustrate this application for the UPI family let's consider the job of controlling the Braemar CM-600 Mini-Dek®.

THE CM-600 MINI-DEK®

The Braemar CM-600 is representative of digital mini-cassette transports. It is a single-head, single-motor transport which operates entirely from a single 5-volt power supply. Its power requirements, including the motor, are 200ma for read or write and 700ma for rewind. Tapes speeds are 3 inches per second (IPS) during read or write, 5 IPS fast forward, and 15 IPS rewind. With these speeds and a maximum recording density of 800 bits per inch (BPI), the maximum data rate is 2400 bits per second (BAUD). The data capacity using both sides of a 100-foot tape is 200K bytes. On top of this, the transport occupies only 22.5 cubic inches (3″x3″x2.5″).

All I/O for the CM-600 is TTL-compatible and can be divided into three groups: motor control, data control, and cassette status. The motor group controls are GO/STOP, FAST/SLOW, and FORWARD/REVERSE. The data controls are READ/WRITE, DATA IN, and DATA OUT. The remaining group of outputs give the transport's status: CLEAR LEADER, CASSETTE PRESENCE, FILE PROTECT, and SIDE SENSOR. These signals, shown schematically in figure 3 and table 1, give the pin definition of the CM-600 16-pin I/O connector.

RECORDING FORMAT

The CM-600 does not provide either encoding or decoding of the recorded data; that task is left for the peripheral interface. A multitude of encoding techniques from which the user may choose are available. In this single-chip dedicated controller application, a "self-clocking" phase encoding scheme similar to that used in floppy discs was chosen. This scheme specifies that a logic "0" is a bit cell with no transition; a cell with a transition is a logic "1."

Table 1. CM-600 I/O Pin Definition

Pin	I/O	Function
1	—	Index pin—not used
2	—	Signal ground
3	O	Cassette side (0—side B, 1—side A)
4	I	Data input (0—space, 1—mark)
5	O	Cassette presence (0—cassette, 1—no cassette)
6	I	Read/Write (0—read, 1—write)
7	O	File protect (0—tab present, 1—tab removed)
8	—	+5v motor power
9	—	Power ground
10	—	Chassis ground
11	I	Direction (0—forward, 1—rewind)
12	I	Speed (0—fast, 1—slow)
13	O	Data output (0—space, 1—mark)
14	O	Clear leader (0—clear leader, 1—off clear leader)
15	I	Motion (0—go, 1—stop)
16	—	+5v logic power

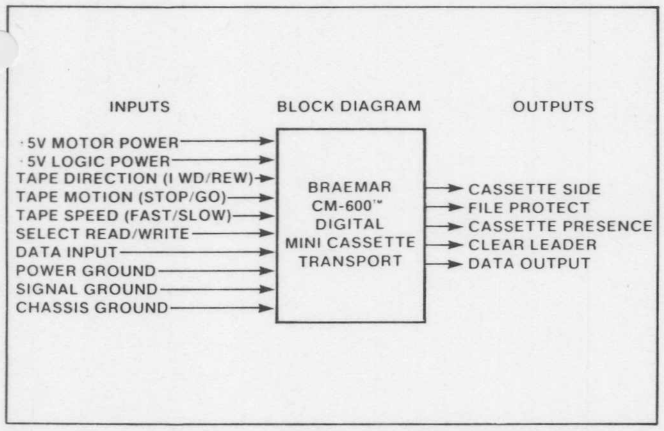

Figure 3. Braemar CM-600™ Block Diagram

Figure 4 illustrates the encoding of the character 3AH assuming the previous data ended with the data line high. (The least significant bit is sent first.) Notice that there is always a "clocking" transition at the beginning of each cell. Decoding is simply a matter of triggering on this "clocking" transition, waiting 3/4 of a bit cell time, and determining whether a mid-cell transition has occurred. Cells with no mid-cell transitions are data 0's; cells with transitions are data 1's. This encoding technique has all the benefits of Manchester encoding with the added advantage that the encoded data may be "decoded by eyeball:" long cells are always 0's, short cells are always 1's.

Besides the encoding scheme, the data format is also up to the user. This controller uses a variable byte length, checksum protected block format. Every block starts and ends with a SYNC character

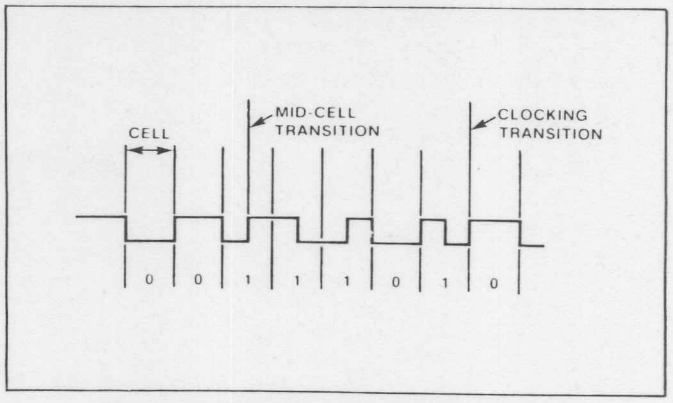

Figure 4. Modified Phase Encoding of Character 3A Hex

(AAH), and the character immediately preceeding the last SYNC is the checksum. The checksum is capable of catching 2 bit errors. The number of data characters within a block is limited to 64K bytes. Blocks are separated by an Inter-Record Gap (IRG). The IRG is of such a length that the transport can stop and start within an IRG, as illustrated in the data block timing, figure 5. Braemar specifies a maximum start or stop time of 150ms for the transport, thus the controller uses 450ms for the IRG. This gives plenty of margin for controlling the transport and also for detecting IRGs while skipping blocks.

THE UPI-41A CONTROLLER

The goal of the UPI software design for this application was to make the UPI-41A microcomputer into an intelligent cassette control processor. The host processor (8086, 8088, 8085A, etc.) simply issues a high-level command such as READ-a-block or WRITE-a-block. The 8741A accepts the command, performs the requested operation, and returns to the host system a result code telling the outcome of the operation, eg. Good-Completion, Sync Error, etc. Table 2 shows the command and result code repertoire. The 8741A completely manages all the data transfers for reading and writing.

As an example, consider the WRITE-a-block command. When this command is issued, the UPI-41A expects a 16-bit number from the host telling how many data bytes to write (up to 64K bytes per block). Once this number is supplied in the form of two bytes, the host is free to perform other tasks; a bit in the UPI's STATUS register or an interrupt output will notify the host when a data transfer is required. The 8741A then checks the transport's status to be sure that a cassette is present and not file protected. If either is false, a result code is

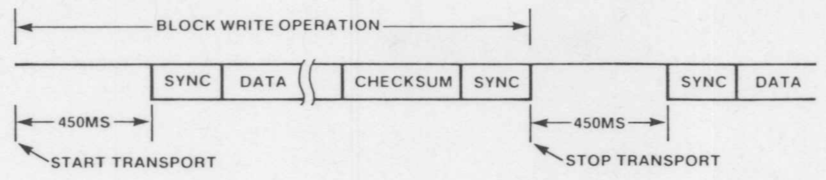

Figure 5. IRG/Block Timing Diagram (not to scale)

Table 2. Controller Command/Result Code Set

Command	Result
Read (01H)	Good-Completion (00H) Buffer Overrun Error (41H) Bad Synch1 Error (42H) Bad Synch2 Error (43H) Checksum Error (44H) Command Error (45H) End of Tape Error (46H)
Rewind (04H)	Good-Completion (00H)
Skip (03H)	Good-Completion (00H) End of Tape Error (47H) Beginning of Tape Error (48H)
Write (02H)	Good-Completion (00H) Buffer Underrun Error (81H) Command Error (82H) End of Tape Error (83H)

returned to the host; otherwise the transport is started. After the peripheral controller checks to make sure that the tape is off the clear leader and past the hole in the tape, it writes a 450ms IRG, a SYNC character, the block of data, the checksum, and the final SYNC character. (The tape has a clear leader at both ends and a small hole 6 inches from the end of each leader.) The data transfers from the host to the UPI-41A slave microcomputer are double buffered. The controller requests only the desired number of data bytes by keeping track of the count internally.

If nothing unusual happened, such as finding clear leader while writing, it returns a Good-Completion result code to the host. If clear leader was encountered, the transport is stopped immediately and an End-of-Tape result code is returned to the host. Another possible error would be if the host is late in supplying data. If this occurs, the controller writes an IRG, stops the drive, and returns the appropriate Data-Underrun result code.

The READ-a-block command also provides error checking. Once this command is issued by the host, the controller checks for cassette presence. If present, it starts the transport. The data output from the transport is then examined and decoded continuously. If the first character is not a SYNC, that's an error and the controller returns a Bad-First-SYNC result code (42H) after advancing to the next IRG. If the SYNC is good, the succeeding characters are read into an on-chip 30 character circular buffer. This continues until an IRG is encountered. When this occurs, the transport is stopped. The controller then tests that the last character. If it is a SYNC, the controller then compares the accumulated internal checksum to the block's checksum, the second to the last character of the block. If they match, a Good-Completion result code (00H) is returned to the host. If either test is bad, the appropriate error result code is returned. The READ command also checks for the End-of-Tape (EOT) clear leader and returns the appropriate error result code if it is found before the read operation is complete.

The 30 character circular buffer allows the host up to 30 character times of response time before the host must collect the data. All data transfers take place thru the UPI-41A Data Bus Buffer Output register (DBBOUT). The controller continually monitors the status of this register and moves characters from the circular buffer to the register whenever it is empty.

The SKIP-n-blocks command allows the host to skip the transport forward or backward up to 127 blocks. Once the command is issued, the controller expects one data byte specifying the number of blocks to skip. The most significant bit of this byte selects the direction of the skip (0=forward, 1=reverse). SKIP is a dual-speed operation in the forward direction. If the number of blocks to skip is greater than 8, the controller uses fast-forward (5 IPS) until it is within 8 blocks of the desired location. Once within 8 blocks, the controller switches to the normal read speed (3 IPS) to allow accurate placement of the tape. The reverse skip uses only the rewind speed (15 IPS). Like the READ and WRITE commands, SKIP also checks for EOT and beginning-of-tape (BOT) depending upon the tape's direction. An error result code is returned if either is encountered before the number of blocks skipped is complete.

The REWIND command simply rewinds the tape to the BOT clear leader. The ABORT command allows the termination of any operation in progress, except a REWIND. All commands, including ABORT, always leave the tape positioned on an IRG.

THE HARDWARE INTERFACE

There's hardly any hardware design effort required for the controller and transport interface in figure 6. Since the CM-600 is TTL compatible, it connects directly to the I/O ports of the UPI controller. If the two are separated (i.e. on different PC cards), it is recommended that TTL buffers be provided.) The only external circuitry needed is an LED driver for the DRIVE ACTIVE status indicator.

The 8741A-to-host interface is equally straightforward. It has a standard asynchronous peripheral interface: 8 data lines (D_0-D_7), read (RD), write (WR), register select (AO), and chip select (CS). Thus it connects directly to an 8086, 8088, 8085A, 8080, or 8048 bus structure. Two interrupt outputs are provided for data transfer requests if the particular system is interrupt-driven. DMA transfer capability is also available. The clock input can be driven from a crystal directly or with the system clock (6MHz max). The UPI-41A clock may be asynchronous with respect to other clocks within the system.

This application was developed on an Intel iSBC 80/30 single board computer. The iSBC 80/30 is controlled by an 8085A microprocessor, contains 16K bytes of dual-ported dynamic RAM and up to 8K bytes of either EPROM or ROM. Its I/O complement consists of an 8255A Programmable Parallel Interface, an 8251A Programmable Communica-

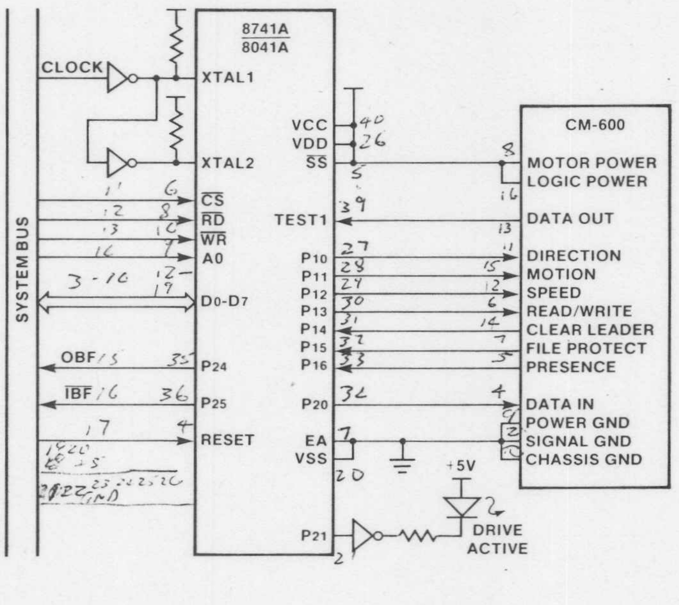

Figure 6. Controller/Transport System Schematic

tions Interface, an 8253 Programmable Interval Timer, and an 8259A Programmable Interrupt Controller. The iSBC 80/30 is especially convenient for UPI development since it contains an uncommitted socket dedicated to either an 8041A or 8741A, complete with buffering for its I/O ports. The iSBC 80/30 to 8741A interface is reflected in figure 8. (Optionally, an iSBC 569 Digital Controller board could be used. The iSBC 569 board contains three uncommitted UPI sockets with an interface similar to that in figure 8.)

Looking at the host-to-controller interface, the host sees the 8741A as three registers in the host's I/O address space: the data register, the command register, and the status register. The decoding of these registers is shown in figure 7. All data and commands for the controller are written into the Data Bus Buffer Input register (DBBIN). The state of the register select input, AO, determines whether a command or data is written. (Writes with AO set to 1 are commands by convention.) All data and results from the controller are read by the host from the Data Bus Buffer Output register (DBBOUT).

$\overline{CS}$	$\overline{RD}$	$\overline{WR}$	A0	Register
0	0	1	0	DBBOUT
0	0	1	1	STATUS
0	1	0	0	DBBIN (DATA)
0	1	0	1	DBBIN (COMMAND)
1	X	X	X	NONE

Figure 7. 8741A/8041A Interface Register Decoding

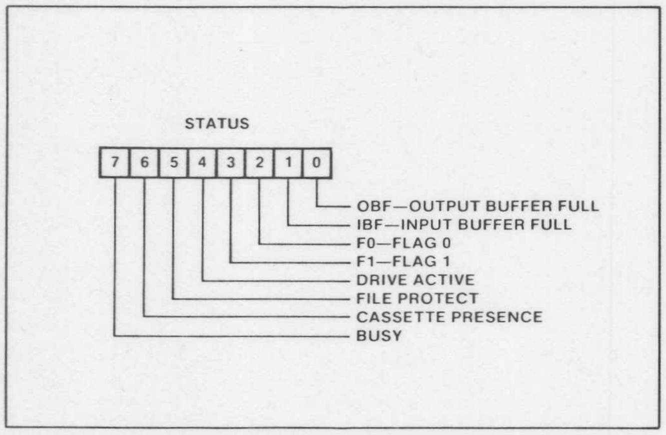

Figure 8. Status Register Bit Definition

The Status register contains flags which give the host the status of various operations within the controller. Its format is given in figure 8. The Input Buffer Full (IBF) and Output Buffer Full (OBF) flags show the Status of the DBBIN and DBBOUT registers respectively. IBF indicates when the DBBIN register contains data written by the host. The host may write to DBBIN only when IBF is 0. Likewise, the host may read DBBOUT only when OBF is set to a 1. These bits are handled automatically by the UPI-41A internal hardware. FLAG 0 (F_0) and FLAG 1 (F_1) are general purpose flags used internally by the controller which have no meaning externally.

The remaining four bits are user-definable. For this application they are DRIVE ACTIVE, FILE PROTECT, CASSETTE PRESENCE, and BUSY flags. The FILE PROTECT and CASSETTE PRESENCE flags reflect the state of the corresponding I/O lines from the transport. DRIVE ACTIVE is set whenever the transport motor is on and the controller is performing an operation. The BUSY flag indicates whether the contents of the DBBOUT register is data or a result code. The BUSY flag is set whenever a command is issued by the host and accepted by the controller. As long as BUSY is set, any character found in DBBOUT is a result code. Thus whenever the host finds OBF set, it should test the BUSY flag to determine whether the character is data or a result code.

Notice the OBF and $\overline{IBF}$ are available as interrupt outputs to the host processor, figure 6. These outputs are self-clearing, that is, OBF is set automatically upon the controller loading DBBOUT and cleared automatically by the host reading DBBOUT. Likewise $\overline{IBF}$ is cleared to a 0 by the host writing into DBBIN: set to a 1 when the controller reads DBBIN into the accumulator.

The flow charts of figure 9 show the flow of sample host software assuming a polling software interface between the host and the controller. The WRITE command requires two additional count bytes which form the 16-bit byte count. These extra bytes are "handshaked" into the controller using the IBF flag in the STATUS register. Once these bytes are written, the host writes data in response to IBF being cleared. This continues until the host finds OBF set indicating that the operation is complete and reads the result code from DBBOUT. No testing of BUSY is needed since only the result code appears in the DBBOUT register.

The READ command does require that BUSY be tested. Once the READ command is written into the

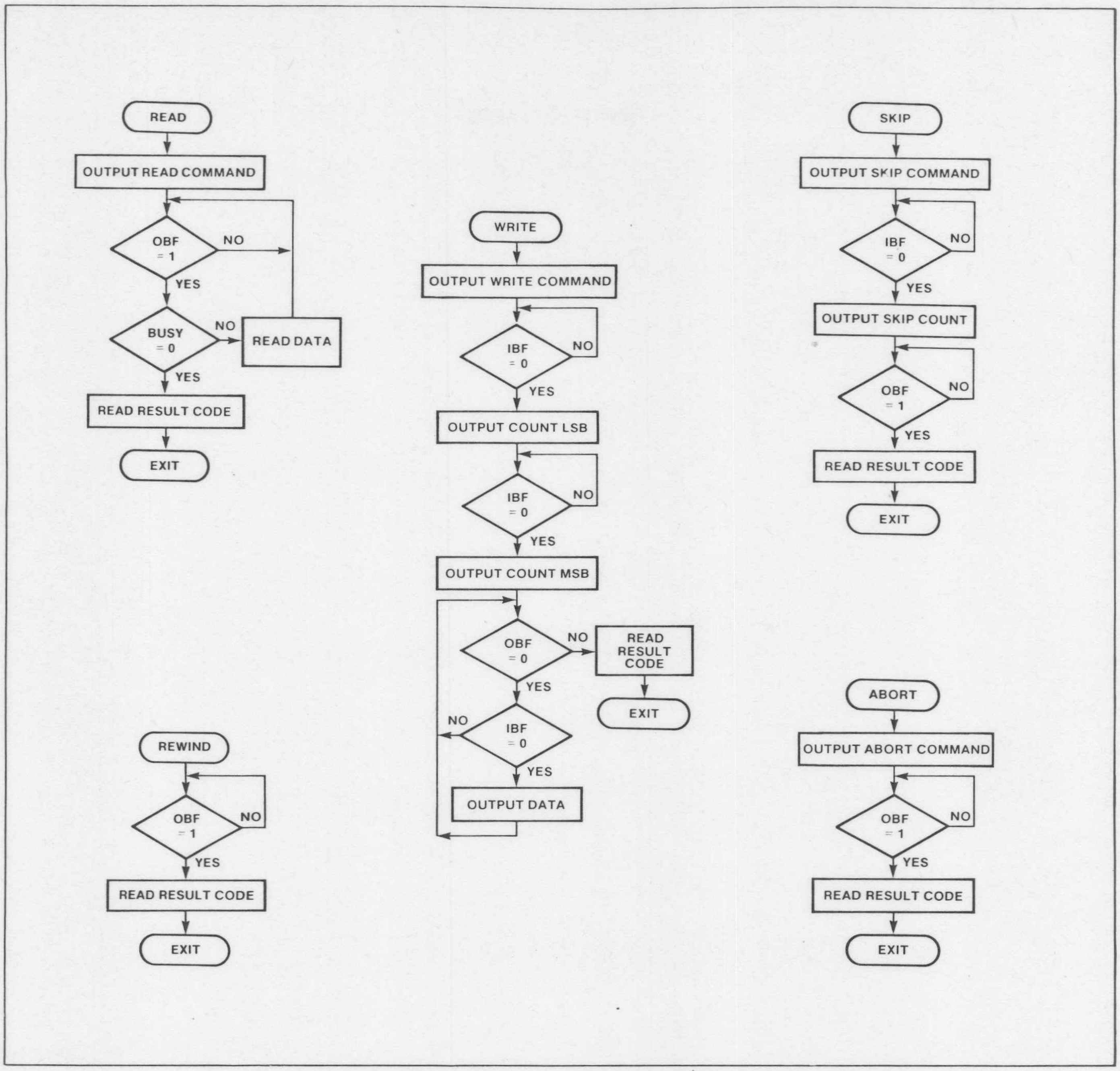

Figure 9. Host CPU Flow Charts for Commands When Polling is Used

controller, the host must test BUSY whenever OBF is set to determine whether the contents of DBBOUT is data from the tape or the result code.

The SKIP command requires the skip count byte. This byte is written into DBBIN after IBF has been cleared following the command. The host then waits until OBF is set indicating the operation is complete and the result code is waiting in DBBOUT. The REWIND and ABORT commands only require that the host test OBF. Once set, the result code is ready in DBBOUT.

The flow charts for an interrupt-driven system are simplified since no testing of OBF or IBF is required. The mere fact that an interrupt occurred implies that the corresponding bit in the STATUS REGISTER is set or cleared.

THE CONTROLLER SOFTWARE

The internal UPI-41A software can be divided roughly into the various commands. (This software is discussed as flow charts. The actual program listing is included in Appendix A.) A command recognizer simply waits for a command input by the host and then branches to the appropriate command routine. The command routine executes until the entire operation is complete and then branches back to the command recognizer. Since only one command routine is executing at any one time, the working registers change function based upon which command is active. Figure 10 shows the register function and identifying name for each command. Notice that while most registers have completely different meanings depending upon the command, some registers retain their meaning over all commands. All registers were assigned names based on their function to aid programming and to make the listing easier to read.

The READ and WRITE commands utilize the internal timer and event counter for all bit timing. This timer provides an internal interrupt on overflow. Thus these commands can be thought of as containing both foreground and background (interrupt service routine) tasks. These tasks communicate via general purpose registers assigned the function of internal status registers: WSTAT and RSTAT for the WRITE and READ commands respectively. The bit definition for these internal status registers is shown in figure 11. We will refer to these bits as the command routines are discussed.

WRITE COMMAND

• REGISTER BANK 0

R0	CNTLSB	BYTE COUNT LSB
R1	CNTMSB	BYTE COUNT MSB
R2	CMPSAV	COMMAND SAVE
R3	CHKSUM	CHECKSUM ACCUMULATOR
R4	TEMP1	TEMPORARY STORAGE
R5		SOFTWARE DELAY COUNTERS
R6		SOFTWARE DELAY COUNTERS
R7	STAT	IMAGE OF UPPER 4-BITS OF STATUS

• REGISTER BANK 1

R0		NOT USED
R1		NOT USED
R2	RESULT	RESULT CODE STORAGE
R3	WSTAT	WRITE STATUS
R4	BITCNT	BIT COUNTER
R5	SERIAL	SERIALIZER
R6	TEMP0	TEMPORARY STORAGE
R7	ASAVE	ACCUMULATOR STORAGE

Figure 10A. Register Definition for WRITE

READ/SKIP/REWIND COMMANDS

• REGISTER BANK 0

R0	LBOUT	BUFFER OUTPUT POINTER
R1	LBRDY	BUFFER READY POINTER
R2		NOT USED
R3	CHKSUM	CHECKSUM ACCUMULATOR
R4	BLKCNT	BLOCK COUNTER FOR SKIP
R5	BLKTIM	COUNTER TO TIME IRG DURING SKIP
R6	BLKSAV	BACKUP FOR BLKTIM
R7	STAT	IMAGE OF UPPER 4-BITS OF STATUS

• REGISTER BANK 1

R0	LBIN	BUFFER INPUT POINTER
R1	IRGCNT	COUNTER TO TIME IRG DURING READ
R2	RESULT	RESULT CODE STORAGE
R3	RSTAT	READ STATUS REGISTER
R4	BITCNT	BIT COUNTER
R5	DESERL	DE-SERIALIZER
R6	RDATA	READ DATA HOLDING REGISTER
R7	ASAVE	ACCUMULATOR STORAGE

Figure 10B. Register Definition for READ, SKIP, and REWIND

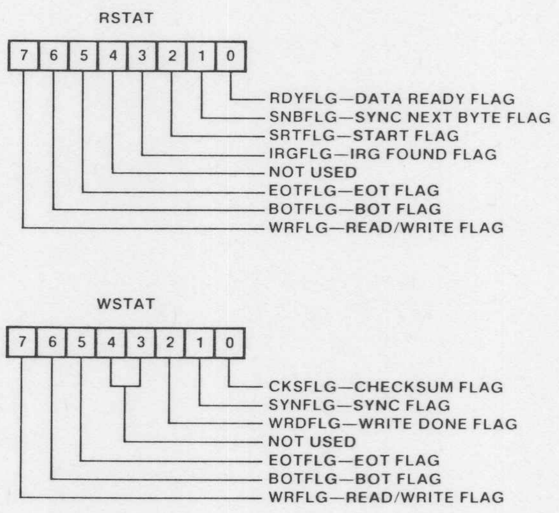

Figure 11. READ and WRITE Internal Status Register Bit Definitions

WRITE COMMAND

Let's look at the WRITE command routine first, figure 12. As was mentioned earlier, the WRITE requires two additional data bytes before it can be processed. Once the command recognizer branches to the WRITE routine, the routine waits on IBF until these bytes are written by the host. These count bytes are stored in the CNTLSB and CNTMSB (Count Least and Most Significant Byte) registers. These two registers are concatenated to form the 16-bit byte count. At this point, the routine tests the transport status lines, CASSETTE PRESENCE and FILE PROTECT. If there is no cassette present or the tape is write protected, the routine exits immediately after resetting BUSY and loading DBBOUT with the appropriate error result code. Assuming the transport status is correct, the other registers required by the routine are initialized: the bit counting register (BITCNT) is set to 8; the checksum accumulator (CHKSUM) is cleared; the data holding register (SERIAL) is loaded with the first SYNC character. The internal timer counter is then loaded with a value which will cause an internal timer interrupt in one half of a bit-cell time, but not activated.

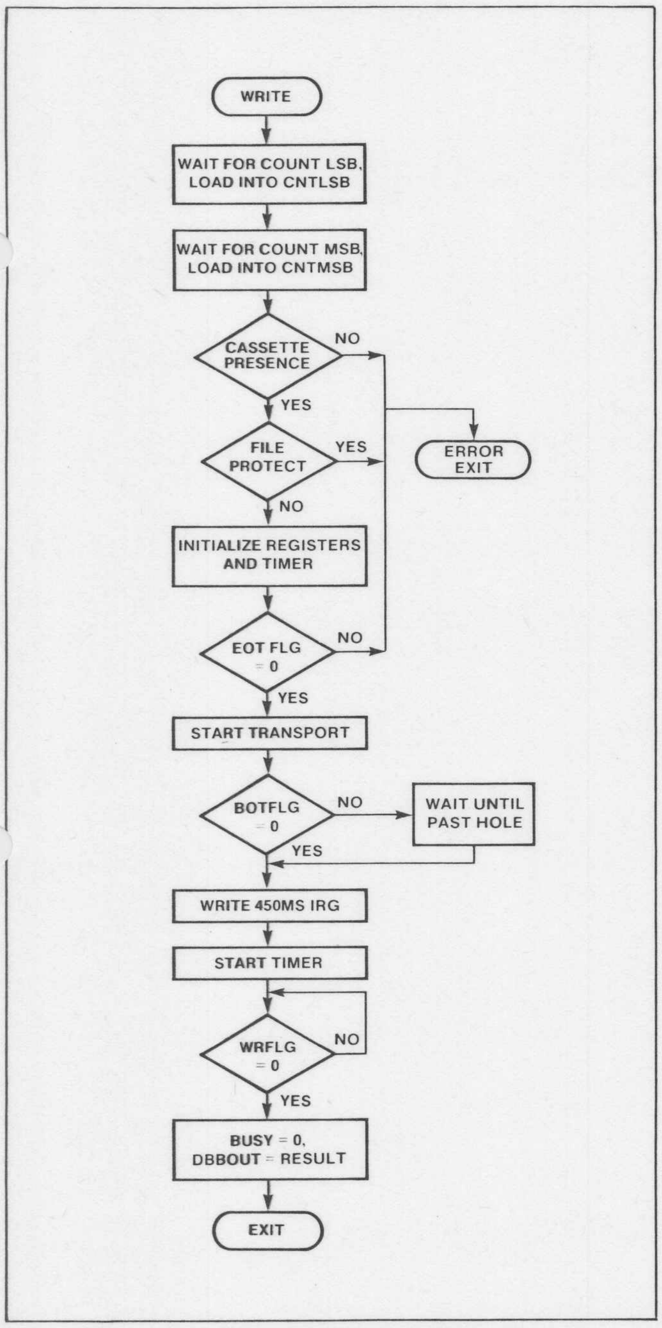

Figure 12. WRITE Command Flow Chart

Next, the EOT flag in WSTAT is examined to see if we are trying to write while at the end of the tape. (EOTFLG is set to 1 if EOT was encountered during the last operation.) If an error occurred, the routine exits after resetting BUSY and loading DBBOUT with the EOT-while-write result error code (83H) via the result storage register, RESULT. Assuming EOTFLG is not set, the DRIVE ACTIVE flag in the Status register is set and the transport is started. The BOT flag (BOTFLG) in WSTAT is then tested to see if we are at the beginning of the tape.) If BOTFLG is 0, the routine writes a 450ms IRG using a software delay loop. If BOTFLG is 1, the routine waits until the clear leader and hole in the tape are passed before starting the IRG. WSTAT is then loaded with 80H. This resets EOTFLG and BOTFLG and sets the write and read flag, WRFLG. WRFLG tells the interrupt routines that a write operation is active. As we shall see, the interrupt routine tells the foreground task that the write operation is complete by resetting WRFLG. At this point the routine starts the timer and enters a loop continually testing WRFLG. If WRFLG is 1, the routine simply loops.

Now let's look at the write routine that does all the work: the write timer interrupt service routine. When the timer interrupt occurs half a bit-cell time later, an automatic vector to the INT routine is performed (location 07H in program memory). INT test WRFLG to see whether it's a read or write operation in progress and branches accordingly. Since we are talking about a write operation, the branch is to the WRINT routine, figure 13. WRINT first reloads the timer to provide the timing for the next half cell (the timer continues to run). The F_0 is used to define whether this particular interrupt is for the first or the second half of the bit cell. The phase encoding algorithm used specifies that the beginning of a bit must always have a transition. If F_0 is reset, the data output to the transport is simply

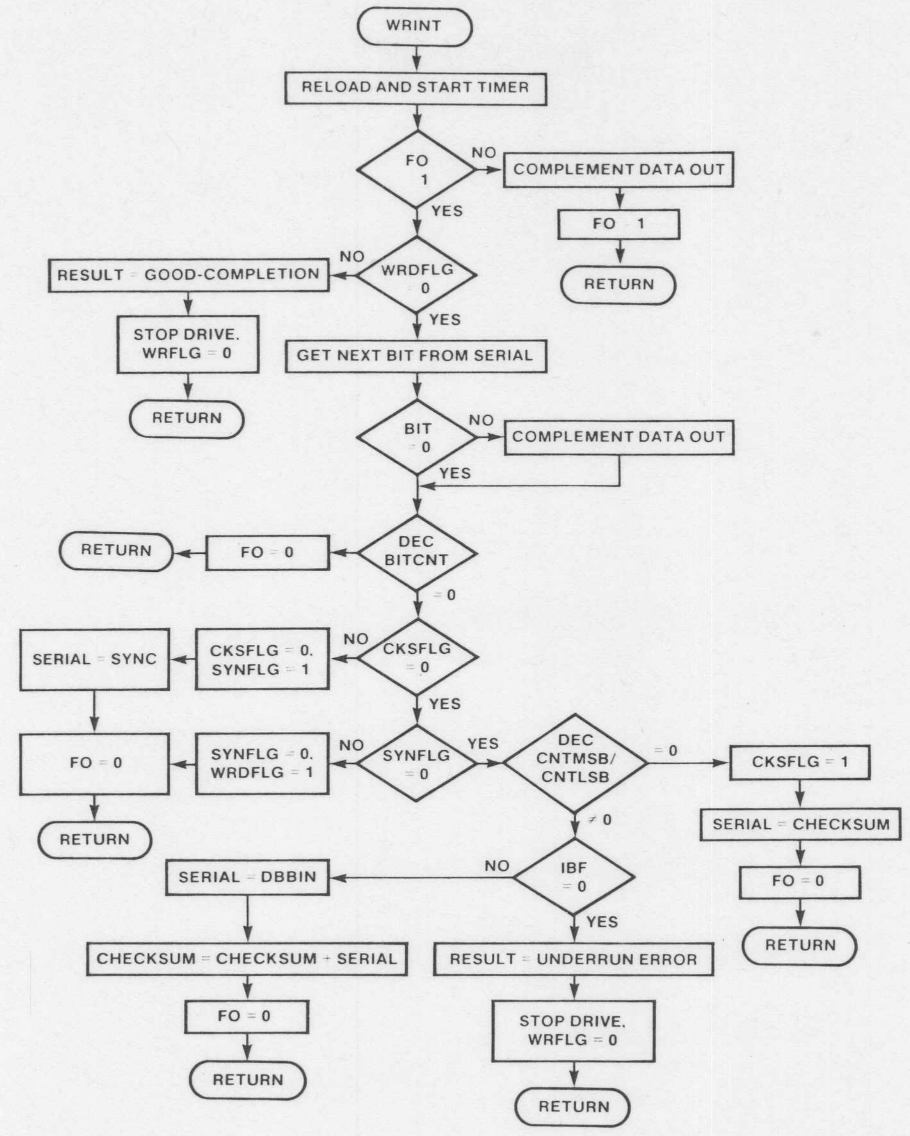

Figure 13. WRINT—Write Timer Interrupt Routine Flow Chart

complemented providing the transition. If set, the interrupt is at the mid-cell position. If the data bit is a 1, complement the data output; otherwise, do not change it. F_0 is complemented every interrupt.

The CLEAR LEADER input from the transport is also tested on every interrupt. If it was encountered, the transport is stopped, the EOTFLG in WSTAT is set, WRFLG is reset, and RESULT is loaded with the EOT-while-write error result code (83H). WRINT returns to the main write loop.

The data contained in the SERIAL register is shifted out bit-by-bit at every other timer interrupt (those interupts with F_0 set to a 1) until the BITCNT register indicates that all 8 bits have been shifted out. When this occurs, a 16-bit decrement operation on the CNTMSB and CNTLSB registers is performed. If the result is non-zero, the routine transfers the next data byte from DBBIN to SERIAL. If the host is late in geting the next byte into DBBIN, a Write-Underrun error (81H) occurs. Like the other error conditions, WRFLG in WSTAT is reset and the Write-Underrun error result code is loaded into the result holding register, RESULT, before returning to the main write loop. If the data is ready in DBBIN, it is transferred to SERIAL and added to the accumulating checksum. The routine then returns to the write main foreground task. (Remember that the foreground task is doing nothing more than testing WRFLG.)

If the decrement result is zero, all data transfers are complete. The accumulated checksum value is loaded into Serial and the Checksum flag, CKSFLG, is set in WSTAT before exiting the interrupt routine. This causes the checksum value to be written onto the tape. Sixteen timer interrupts later the checksum is complete; it is now time to write the final SYNC. CKSFLG is reset, a SYNC character is loaded into SERIAL, and the SYNC flag (SYNFLG), is set in WSTAT. Sixteen more timer interrupts later the SYNC is written to the tape and the block is almost finished. One more interrupt is needed to finish the last bit. The write done flag (WRDFLG) is set to indicate that this is the last interrupt for this block. WRDFLG is detected as being set to a 1 on the next interrupt and the transport is stopped. WRFLG in WSTAT is reset and the Good-Completion result code is loaded into the RESULT register before exiting to the foreground task.

All this occurs while the foreground task is testing WRFLG. When WRFLG is cleared, the foreground task "knows" that the background task is finished; BUSY is reset and the result code stored in RESULT is loaded into DBBOUT. The program then returns to the command recognizer.

READ OPERATION

In the case of the read command, figure 14, the RSTAT register provides the communication between the foreground and background tasks. The read command routine starts out by initializing the registers it requires: the checksum accumulator, CHKSUUM, is cleared; pointers for the circular buffer, DBIN, LBRDY, and LBOUT are set to the start of the buffer; and the bit counter, BITCNT, is set to 8.

The circular buffer has three pointers: LBIN to point to the next free buffer location, LBOUT to point to the next location from which to retrieve data, and LDRTDY to trail LBIN by two locations. LBRDY trails LBIN to ensure that the host does not get the received checksum or last SYNC bytes as data. The buffer is empty whenever LDRDY equals LBOUT. The buffer is full whenever LBOUT minus 1 equals LBIN. Data is placed in the buffer by loading it into the location pointed at by LBIN and then LBIN is incremented. Data is removed from the buffer at the location pointed at by LBOUT and the LBOUT is incrementd by one. The data memory locations 20H thru 3FH form the circular buffer. Any pointer increment or decrement operation limits the pointers to this range. (If a pointer at 3FH is incremented, the result wraps around to 20H.)

Once the registers have been initialized, the timer is loaded, but not started, with a value that corresponds to 3/4 of a bit-cell time. Next, the EOT test is performed on the EOTFLG in RSTAT. The routine exits with an EOT-while-read error result code if an attempt to read is made while at EOT. If not, the transport is started and the BOTFLG is tested to see if it must move past the clear leader and hole. Once past the clear leader and hole, if necessary, the SYNC-Next-Byte flag (SNBFLG), and the Start flag (SRTFLG), are set in RSTAT. SNBFLG informs the software that the next received byte should be a SYNC. SRTFLG prevents LBRDY from being incremented prematurely.

As soon as a transition from mark (1) to space (0) is detected, the timer is started. The routine enters a loop which tests the data ready flag (RDYFLG), the IRG found flag (IRGFLG), and the EOT detected flag (EOTFLG) in RSTAT. These flags are set by the background task to communicate with the foreground. RDYFLG is set when a character has been assembled and is waiting in the holding register, RDATA. IRGFLG is set when an IRG has been found by the background task. EOTFLG has the same meaning as with the write command; the clear leader at the end of the tape has been found.

If none of these flags are set, the foreground then looks at the circular buffer to see if it contains any data to output to the host. The buffer contains data when LBIN does not equal LBOUT. If these pointers are equal, the buffer is empty and the foreground task just continues to loop. If they are not equal, there is some data left in the buffer. OBF is tested to see if DBBOUT is free to accept more data. If it is free, the character pointed at by LBOUT is transfered to DBBOUT and LBOUT is incremented to the next location. If DBBOUT still contains previously loaded data (OBF set), the foreground continues to test the flags in RSTAT.

When the foreground task finds RDYFLG set, data is available in RDATA. Before transfering this data into the buffer, it first compares LBIN and LBOUT. If LBOUT is one less than LBIN, the buffer is full and no more data can be loaded. This is an error condition; the read operation is aborted and the

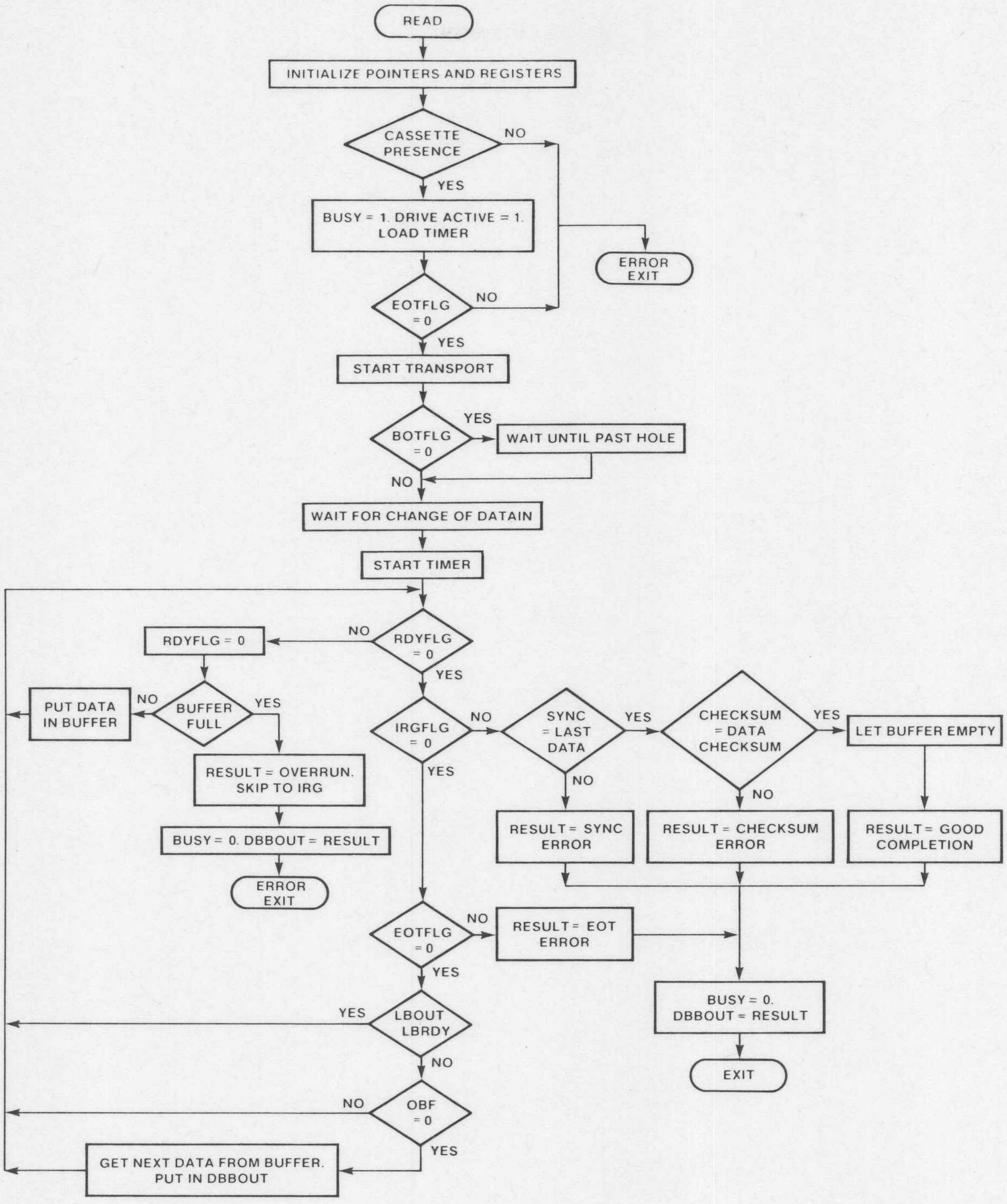

Figure 14. READ Command Routine Flow Chart

transport is moved to the next IRG using the SKIP routine discussed below. Once at the next IRG, BUSY is reset and the Read-Overrun error result code (41H) is placed in DBBOUT. This terminates the read operation and the routine branches back to the command recognizer.

If the buffer is not full, the data is transfered from RDATA to the location pointed to by LBIN. LBIN is incremented and the RDYFLG in RSTAT is reset. LBRDY is also incremented if LBIN has been incremented twice already. (SRTFLG set prevents LBRDY from being incremented. SRTFLG is reset when LBIN is incremented to the second buffer position.) This ensures that LBRDY will point to the last data byte once an IRG is detected. The data is also added to the accumulated checksum, CHKSUM. The foreground then goes back to test the RSTAT flags. When IRGFLG is found set, the background task has found an IRG and stopped the transport. This indicates that the block read is complete. Since the IRG occurs after the checksum and final SYNC characters, these two bytes are in the circular buffer. To test them the foreground task then decrements LBIN to point at the final SYNC and checks if it is a SYNC character. If not, a Bad-Sync2 error result code (43H) is placed in RESULT and the routine branches to the read exit routine. If it is okay, a SYNC is removed from the accumulated checksum. LBIN is decremented again to point to the received checksum. Since this character is also in the accumulated checksum, it is subtracted out. Now the accumulated checksum reflects only the received data so it is compared with the received checksum. If they are equal, the data is presumed good and a Good-Completion result code (00H) is loaded into result. If not, an error has occured and the RESULT is loaded with the Bad-Checksum result code (44H).

Although the actual read operation is complete with respect to the transport, there may still be data remaining in the buffer of the controller. The read exit routine loops testing LBOUT and LBRDY and transfering data from the buffer into DBBOUT until the buffer is empty. Once the buffer is empty, BUSY is reset and the result code is transferred from RESULT to DBBOUT, completing the read operation.

The timer interrupt routine, RDINT, for the read operation is shown in figure 15. The phase decoding algorithm specifies that the timer start at the beginning transition of the bit cell. It waits for 3/4 of a bit cell before sampling the data input. If the data input is the same as immediately after the beginning transition, the data bit is a 0. If it is different, the data bit is a 1. The timer interrupt routine compares the present state of the data input to the state immediately following the beginning transition. F_0 stores this value and shifts it into the de-serializing register (DESERL). Once 8 bits have been accumulated, the RDYFLG is set to inform the foreground that a character is complete. This character is then transfered from DESERL to the holding register, RDATA.

After the interrupt routine has sampled and shifted in the bit, it looks for the beginning transition of the

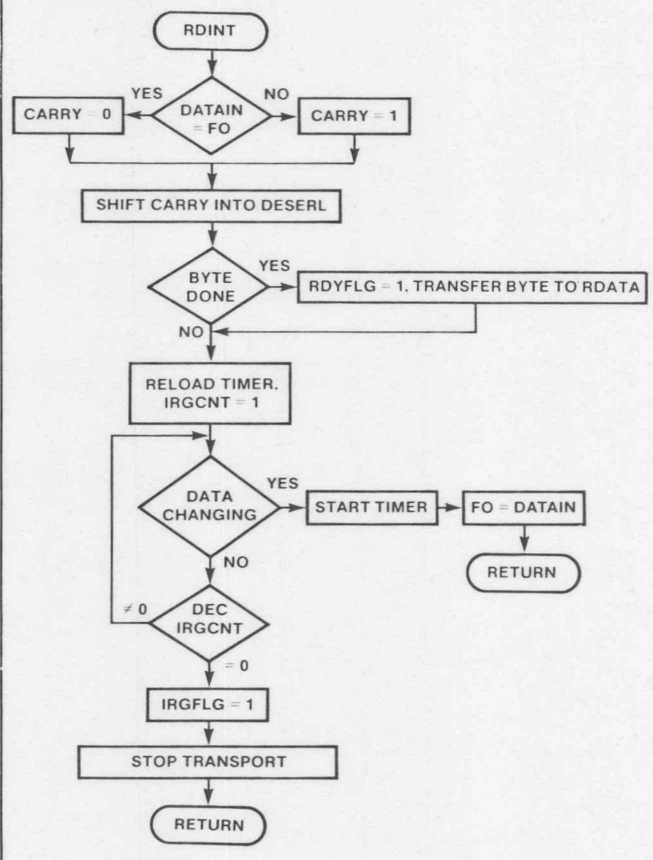

Figure 15. RDINT—Read Timer Interrupt Routine Flow Chart

next bit cell. While looking for this transition, it keeps track of time be decrementing a counter called IRGCNT. If this counter reaches zero, no transition has occured within a certain amount of time (this application used two bit cell times); this is defined as the beginning of an IRG. When an IRG is found, the transport is stopped and the IRGFLG is set in RSTAT before exiting the interrupt service routine. If a transition is found before the counter times out, the routine exits setting F_0 to the data input state after the transition. F_0 is used for storing the state while in the foreground. As in the write operation, the CLEAR LEADER input is also tested every interrupt. If an EOT is found, EOTFLG is set and the transport is stopped.

SKIP OPERATION

The same technique for finding IRGs is used in the SKIP command routine. The SKIP command, figure 16, causes the transport to skip forward or reverse a specified number of IRGs. The number of IRGs to skip is indicated by the byte following the SKIP command byte acceptance (i.e. BUSY has been set and IBF is 0). The SKIP command routine waits, looping on IBF, until the IRG skip count is loaded by

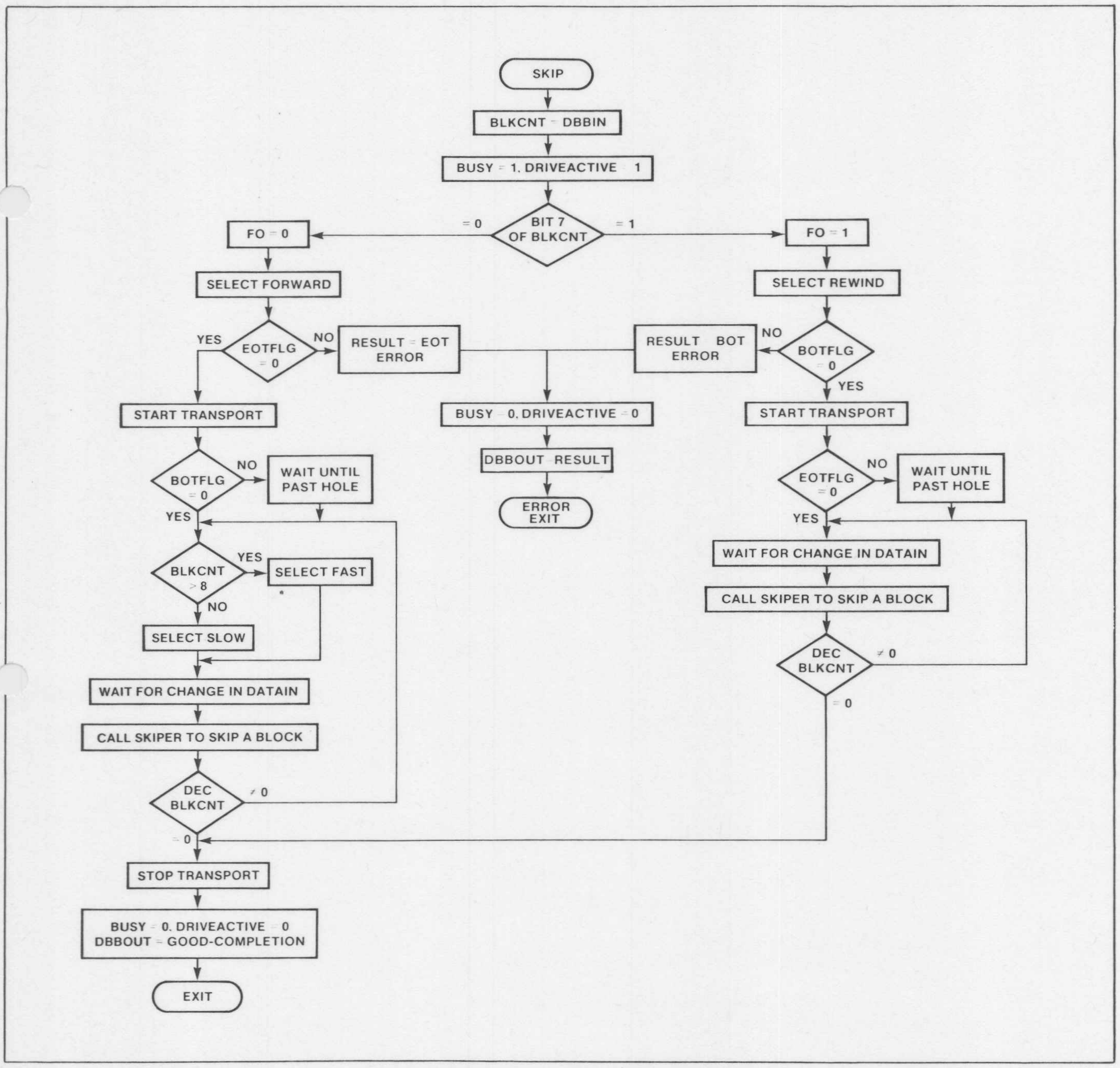

Figure 16. SKIP Command Routine Flow Chart

the host. Then it is transferred from DBBIN to the skip count register, BLKCNT. The usual EOT and BOT tests are performed to ensure it doesn't skip forward when at EOT or reverse when at BOT. The transport is then started in the direction indicated by bit 7 of the BLKCNT value. (This bit is masked off after the initial direction test.)

For reverse skips, the skipping subroutine, SKIPER, is called. It advances the transport to the next IRG using the IRGCNT technique described above. When SKIPER returns, BLKCNT is decremented and tested for zero. If non-zero, SKIPER is called repeatedly until BLKCNT is zero. Once zero, the transport is stopped and BUSY is reset. DBBOUT is loaded with a Good-Completion result code (00H).

When doing forward skips, the software takes advantage of the fact that the transport can recognize IRGs during fast forward. If the BLKCNT is greater than 8, fast forward is selected instead of slow and SKIPER is called. (The IRGCNT value is modified to take into account the faster tape speed.) When SKIPER returns, BLKCNT is decremented and tested both for being less than 8 or equal to zero. Once BLKCNT is less than 8, the slow speed is selected. Once BLKCNT reaches 0, the operation is terminated like the reverse skips. The transport is stopped and BUSY is cleared. DBBOUT is loaded with a Good-Completion result code.

As with both READ and WRITE commands, the clear leader test is made periodically to ensure that no skips are made past the end or beginning of the tape. The appropriate error result code is issued if CLEAR LEADER is found set. RSTAT is loaded with the appropriate EOTFLG bit set.

REWIND OPERATION

The REWIND command routine, figure 17, simply sets the transport to fast rewind and loops until clear leader is found for greater than 50ms. (The hole at the ends of the tape is guaranteed not to cause the clear leader input to be active for more than 50ms.) Once the tape's clear leader is found; the transport is stopped; BUSY is reset. A Good-Completion result code is loaded into DBBOUT. Also, since the transport is now at the BOT, the BOTFLG is RSTAT is set.

ABORT OPERATION

The final command is the ABORT command. It does not have a separate flow chart of its own. All other commands monitor IBF periodically during their execution. If a command is found, the command is compared to the ABORT command code. If it is found, the routine in execution is stopped and BUSY is reset. The Abort-Complete result code is placed in DBBOUT. The aborted routine does ensure that it exits gracefully. An aborted READ or SKIP advances to the next IRG before stopping; WRITE records an IRG before stopping.

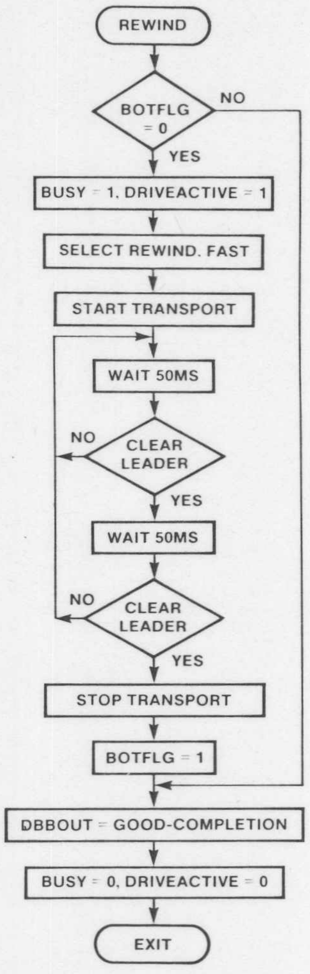

Figure 17. REWIND Command Routine Flow Chart

WRAPPING IT UP

The program listing follows in Appendix A. For more information on the UPI-41A family, see the referenced manuals on the cover of this application note. For those readers who would like to use or modify this program but don't want to type in nearly 1K bytes of code, source files are available through the Intel User's Library, INSITE. (Contact your local Intel sales office for information on INSITE.) A sample of other UPI-41A programs available thru the INSITE library are:

Seiko printer controller
Olivetti printer controller
LRC printer controller (8295)
Sensor matrix controller
LED display controller
Combination serial/parallel I/O
Programmable keyboard/display controller
GPIB controller (8292)

```
ASM48 :F1:DIGCAS.ASM NOOBJECT PRINT(:LP:)


ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

  LOC  OBJ          LINE        SOURCE STATEMENT

                       1 $MACROFILE MOD41A TITLE('DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80')
                       2 ;
                       3 ;**********************************************************************
                       4 ;
                       5 ;UPI-41A DIGITAL CASSETTE CONTROLLER FOR THE BRAEMAR CM-600
                       6 ;
                       7 ;**********************************************************************
                       8 ;
                       9 ;THIS UPI-41A BASED PROGRAM CONTROLS A BRAEMAR CM-600 MINI-CASSETTE.
                      10 ;THE PROGRAM ALLOWS THE HOST CPU TO SIMPLY ISSUE COMMANDS SPECIFYING
                      11 ;READ-A-BLOCK, WRITE-A-BLOCK, SKIP FORWARD OR REVERSE N BLOCKS,
                      12 ;REWIND, AND ABORT.  THE UPI-41A HANDLES ALL DATA REQUESTS AND MONITORS
                      13 ;THE CASSETTE DRIVE FOR ERRORS; EG. WRITING TO THE END-OF-TAPE, ETC.
                      14 ;EACH COMMAND SETS THE CONTROLLER IN THE BUSY CONDITIONS.  ONCE THE
                      15 ;OPERATION IS COMPLETE, THE UPI-41A RESETS IT'S BUSY FLAG AND LOADS THE
                      16 ;OUTPUT DATA BUFFER WITH A RESULT BYTE WHICH INDICATES THE RESULT
                      17 ;OF THE REQUESTS OPERATION.  THE COMMANDS AND RESULT CODES ARE SHOWN
                      18 ;IN THE SYSTEMS EQUATES.
                      19 ;
                      20 ;THE CONTROLLER USES A MODIFIED PHASE ENCODING WHERE DATA 0'S ARE LONG
                      21 ;(FULL BIT TIME) CELLS AND DATA 1'S HAVE TRANSITIONS AT THE MID-BIT CELL
                      22 ;POSITION.  WHEN WRITING, ALL BLOCKS ARE PREFACED AND CONCLUDED WITH
                      23 ;SYNC CHARACTERS (0AAH).  A CHECKSUM BYTE IMMEDIATELY PRECEEDS THE
                      24 ;FINAL SYNC.  WHEN READING, THE CONTROLLER TESTS THE VALIDITY OF
                      25 ;BOTH SYNC CHARACTERS AND THE CHECKSUM.
                      26 ;INTER-RECORD GAPS (IRG) ARE WRITTEN WITH ALL MARK (DATA OUT = 1).
                      27 ;
                      28 ;THE WRITE-A-BLOCK OPERATION IS DOUBLE BUFFERED WHILE THE READ-A-BLOCK
                      29 ;OPERATION USES A 30-CHARACTER CIRCULAR BUFFER TO MINIMIZE CPU
                      30 ;RESPONSE TIME REQUIREMENTS.
                      31 ;
                      32 $EJECT
                      33 ;**********************************************************************
                      34 ;
                      35 ;REGISTER EQUATES - THE WRITE AND READ/SKIP OPERATIONS ARE DISTINCT
                      36 ;THEREFORE THE SAME PHYSICAL REGISTER MAY BE USED FOR DIFFERENT
                      37 ;PURPOSES IN EACH OPERATION.  EACH OPERATION USES DIFFERENT REGISTER
                      38 ;LABELS FOR CLARITY.
                      39 ;
                      40 ;**********************************************************************
                      41 ;
                      42 ;        WRITE - RB0
                      43 ;        -----------
  0000                44 CNTLSB  EQU     R0      ;BYTE COUNTER LSB
  0001                45 CNTMSB  EQU     R1      ;BYTE COUNTER MSB
  0002                46 CMDSAV  EQU     R2      ;COMMAND SAVER
  0003                47 CHKSUM  EQU     R3      ;CHECKSUM REGISTER
  0004                48 TEMP1   EQU     R4      ;TEMPORARY STORAGE
                      49 ;               R5       DELAY REGISTER
                      50 ;               R6       DELAY REGISTER
  0007                51 STAT    EQU     R7      ;STS IMAGE
                      52 ;
                      53 ;        WRITE - RB1
                      54 ;        -----------
                      55 ;               R0       NOT USED
                      56 ;               R1       NOT USED
  0002                57 RESULT  EQU     R2      ;RESULT STORAGE
  0003                58 WSTAT   EQU     R3      ;WRITE STATUS REGISTER
  0004                59 BITCNT  EQU     R4      ;WRITE BIT COUNTER
  0005                60 SERIAL  EQU     R5      ;WRITE SERIALIZER
  0006                61 TEMPO   EQU     R6      ;TEMPORARY STORAGE
  0007                62 ASAVE   EQU     R7      ;ACCUMULATOR SAVE
                      63 ;
                      64 ;**********************************************************************
                      65 ;
                      66 ;        READ/SKIP - RB0
                      67 ;        ---------------
  0000                68 LBOUT   EQU     R0      ;NEXT BYTE OUTPUT POINTER
  0001                69 LBRDY   EQU     R1      ;NEXT BYTE AVAILABLE POINTER
                      70 ;               R2       NOT USED
                      71 ;CHKSUM  EQU    R3      ;CHECKSUM REGISTER (SAME FOR WRITE)
  0004                72 BLKCNT  EQU     R4      ;BLOCK COUNTER FOR SKIP
  0005                73 BLKTIM  EQU     R5      ;BLOCK IRG TIMER FOR SKIP
  0006                74 BLKSAV  EQU     R6      ;BLOCK IRG TIMER SAVE
                      75 ;STAT    EQU    R7      ;STS IMAGE (SAME FOR WRITE)
                      76 ;
                      77 ;        READ/SKIP - RB1
                      78 ;        ---------------
  0000                79 LBIN    EQU     R0      ;NEXT BYTE INPUT POINTER
  0001                80 IRGCNT  EQU     R1      ;IRG TICK TIMER
                      81 ;RESULT  EQU    R2      ;RESULT STORAGE (SAME FOR WRITE)
  0003                82 RSTAT   EQU     R3      ;READ STATUS REGISTER
                      83 ;BITCNT  EQU    R4      ;READ BIT COUNTER (SAME FOR WRITE)
  0005                84 DESERL  EQU     R5      ;READ DE-SERIALIZER
  0006                85 RDATA   EQU     R6      ;READ DATA BUFFER
                      86 ;ASAVE   EQU    R7      ;ACCUMULATOR SAVE (SAME FOR WRITE)
                      87 ;
  LOC  OBJ         LINE        SOURCE STATEMENT

;*********************************************************************
;
;STATUS REGISTER BIT DEFINITIONS:
;THE MAJOR OPERATIONS, WRITE AND READ, USE THE TIMER TO DETERMINE
;ALL BIT-CELL TIMING AND TO PERFORM THE SERIAL-TO-PARALLEL CONVERSIONS.
;THE TIMER INTERRUPT SERVICE AND MAIN ROUTINES COMMUNICATE VIA
;GENERAL PURPOSE REGISTERS USED AS STATUS REGISTERS.
;
;STAT IS THE STS REGISTER IMAGE SINCE THE UPI CAN'T READ STS DIRECTLY.
;(ONLY THE HIGH ORDER 4-BITS OF STAT ARE USED.)
;THE LOWER 4-BITS ARE NOT USER-DEFINABLE.
;
;*********************************************************************
;
;WSTAT - WRITE STATUS REGISTER
;
;       WSTAT0  CHECKSUM FLAG (CKSFLG) - CHECKSUM BYTE BEING SENT
;            1  SYNC FLAG (SYNFLG) - FINAL SYNC BYTE BEING SENT
;            2  WRITE DONE FLAG (WRDFLG) - FINAL SYNC IS BEING SENT
;                                           (ENSURES LAST BIT IS COMPLETE)
;            3  NOT USED
;            4  NOT USED
;            5  END OF TAPE FLAG (EOTFLG) - EOT WAS FOUND, TAPE IS NOW AT EOT
;            6  BEGINNING OF TAPE FLAG (BOTFLG) - BOT WAS FOUND, TAPE IS NOW AT BOT
;            7  WRITE/READ FLAG (WRFLG) - WRITE OR READ OPERATION IS ACTIVE
;
;RSTAT - READ STATUS REGISTER
;
;       RSTAT0  DATA READY FLAG (RDYFLG) - NEXT BYTE IS READY IN RDATA
;            1  SYNC NEXT BYTE FLAG (SNBFLG) - NEXT BYTE SHOULD BE A SYNC
;            2  START FLAG (SRTFLG) - BEGINNING OF READ, DON'T INC LBRDY
;                                      UNTIL LBIN=22
;            3  IRG FOUND FLAG (IRGFLG) - IRG WAS FOUND BY TIMER INTERRUPT ROUTINE
;            4  NOT USED
;            5  END OF TAPE FLAG (EOTFLG) - EOT WAS FOUND, TAPE IS NOW AT EOT
;            6  BEGINNING OF TAPE FLAG (BOTFLG) - BOT WAS FOUND, TAPE IS NOW AT BOT
;            7  WRITE/READ FLAG (WRFLAG) - WRITE OR READ OPERATION IS ACTIVE
;
;STAT - STS IMAGE
;
;       STAT0  OBF - OUTPUT BUFFER FULL
;           1  IBF - INPUT BUFFER FULL
;           2  F0 - GENERAL PURPOSE FLAG (USED INTERNALLY)
;           3  F1 - COMMAND/DATA FLAG
;           4  DRIVE ACTIVE - MOTOR ON
;           5  FILE PROTECT - DRIVE STATUS
;           6  CASSETTE PRESENCE - DRIVE STATUS
;           7  BUSY - CONTROLLER PERFORMING OPERATION
;
$EJECT
;*********************************************************************
;
;PORT DEFINITION:
;
;*********************************************************************
;
;PORT10 -  DIRECTION (0-FORWARD, 1-REWIND)
;     11 -  MOTION (0-GO, 1-STOP)
;     12 -  SPEED (0-FAST, 1-SLOW)
;     13 -  READ/WRITE (0-READ, 1-WRITE)
;     14 -  CLEAR LEADER (0-OFF LEADER, 1-ON LEADER)
;     15 -  FILE PROTECT (0-TAB PRESENT, 1-NO TAB
;     16 -  PRESENCE (0-TAPE IN WITH DOOR CLOSED, 1-NO TAPE)
;     17 -  NOT USED
;
;PORT20 -  DATA OUT TO CASSETTE (0-SPACE, 1-MARK)
;     21 -  DRIVE ACTIVE LED (0-ON, 1-OFF)
;     22 -  NOT USED
;     23 -  NOT USED
;     24 -  OBF INTERRUPT OUTPUT
;     25 -  IBF/ INTERRUPT OUTPUT
;     26 -  NOT USED
;     27 -  NOT USED
;
;TEST1 -  DATA IN FROM CASSETTE
;
$EJECT
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

                   ;*****************************************************************
                   ;
                   ;SYSTEM EQUATES:
                   ;
                   ;*****************************************************************
                   ;
                   ;WRITE SYSTEM EQUATES:
                   ;
0001               CKSFLG  EQU     01H     ;CHECKSUM FLAG IN WRITE STATUS
0002               SYNFLG  EQU     02H     ;SYNC FLAG IN WRITE STATUS
0004               WRDFLG  EQU     04H     ;WRITE DONE FLAG IN WRITE STATUS
0020               EOTFLG  EQU     20H     ;EOT FLAG
0040               BOTFLG  EQU     40H     ;BOT FLAG
0080               WRFLG   EQU     80H     ;READ/WRITE FLAG IN WRITE STATUS
0008               WRCNT   EQU     08H     ;WRITE BIT CONSTANT
FFFC               WRTIM   EQU     -4H     ;WRITE TIMER CONSTANT
                   ;
                   ;PORT EQUATES:
                   ;
0001               REWIND  EQU     01H     ;DIRECTION MASKS
00FE               FORWD   EQU     0FEH
0002               STP     EQU     02H     ;START/STOP MASKS
00FD               SRT     EQU     0FDH
0004               SLOW    EQU     04H     ;SPEED MASKS
00FB               FAST    EQU     0FBH
0008               WR      EQU     08H     ;WRITE/READ MASKS
00F7               RD      EQU     0F7H
0001               DOHI    EQU     01H     ;DATA OUTPUT TO DRIVE MASKS
00FE               DOLOW   EQU     0FEH
0002               DAOFF   EQU     02H     ;DRIVE ACTIVE LED MASKS
00FD               DAON    EQU     0FDH
                   ;
                   ;READ SYSTEM EQUATES:
                   ;
0001               RDYFLG  EQU     01H     ;DATA READY FLAG IN READ STATUS
0002               SNBFLG  EQU     02H     ;SYNC NEXT BYTE FLAG IN READ STATUS
0004               STRFLG  EQU     04H     ;START INC READY POINTER FLAG IN READ STATUS
0008               IRGFLG  EQU     08H     ;IRG FOUND FLAG IN READ STATUS
0008               RDCNT   EQU     08H     ;READ TIMER CONSTANT
FFFA               RDTIM   EQU     -6H     ;READ BIT CONSTANT
                   ;
                   ;STS REGISTER EQUATES:
                   ;
0080               BUSY    EQU     80H     ;BUSY BIT
0010               DRACT   EQU     10H     ;DRIVE ACTIVE BIT
0040               TAPIN   EQU     40H     ;TAPE IN DRIVE BIT
0020               FILPRT  EQU     20H     ;FILE PROTECT BIT
                   $EJECT
                   ;GENERAL RESULT CODES
                   ;
0001               ABTCMP  EQU     01H     ;ABORT COMPLETE CODE
0000               GOOD    EQU     00H     ;GOOD RESULT CODE
0002               CMDERR  EQU     02H     ;COMMAND ERROR CODE
0003               NTAPE   EQU     03H     ;NO TAPE ERROR CODE
0004               NWR     EQU     04H     ;FILE PROTECT ERROR CODE
                   ;
                   ;WRITE RESULT CODES
                   ;
0081               UNDERW  EQU     81H     ;UNDERRUN ERROR CODE
0082               WCMDER  EQU     82H     ;COMMAND/DATA ERROR CODE
0083               EOTERR  EQU     83H     ;EOT ERROR CODE
                   ;
                   ;READ RESULT CODES
                   ;
0041               OVERUN  EQU     41H     ;UNDERRUN CODE FOR BUFFER
0042               SYNC1   EQU     42H     ;BAD SYNC1 ERROR CODE
0043               SYNC2   EQU     43H     ;BAD SYNC2 ERROR CODE
0044               BADCHS  EQU     44H     ;BAD CHECKSUM ERROR CODE
0045               RCMDER  EQU     45H     ;COMMAND/DATA ERROR CODE
0046               REOTER  EQU     46H     ;EOT AT READ ERROR CODE
0047               SKPEOT  EQU     47H     ;EOT AT SKIP ERROR CODE
0048               SKPBOT  EQU     48H     ;BOT AT RSKIP ERROR CODE
                   ;
                   ;MISC EQUATES:
                   ;
00AA               SYNC    EQU     0AAH    ;SYNC BYTE
0033               SLWIRG  EQU     51D     ;SLOW IRG COUNT CONSTANT - NO TRANSITION IN 2 BIT TIMES
0020               FASIRG  EQU     32D     ;FAST IRG COUNT CONSTANT
0020               RWDIRG  EQU     32D     ;REWIND IRG COUNT FOR SKIP
                   ;
                   ;COMMANDS
                   ;
0005               ABORT   EQU     05H     ;ABORT COMMAND
0001               RDCMD   EQU     01H     ;READ FROM TAPE COMMAND
0002               WRCMD   EQU     02H     ;WRITE TO TAPE COMMAND
0004               RWCMD   EQU     04H     ;REWIND COMMAND
0003               SKCMD   EQU     03H     ;SKIP BLOCK COMMAND
0000               RESCMD  EQU     00H     ;RESET COMMAND
                   ;
                   $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

 LOC  OBJ          SOURCE STATEMENT

                ;**********************************************************************
                ;
                ;START OF PROGRAM - JUMPS FOR COLD START (RESET) AND TIMER INTERRUPTS
                ;
                ;**********************************************************************
                ;
0000 0409       RESET:  JMP     BEGIN           ;JUMP OVER TIMER VECTOR LOCATION
                ;
0007                    ORG     7H              ;TIMER INTERRUPT VECTOR LOCATION
                ;
0007 6400       TIMINT: JMP     INT             ;JUMP TO TIMER INTERRUPT SERVICE ROUTINE
                ;
                ;**********************************************************************
                ;
                ;PROGRAM START - INITIALIZE STATUS REGISTERS, DRIVE OUTPUTS, AND
                ;WAIT FOR A COMMAND.
                ;
                ;**********************************************************************
                ;
0009 27         BEGIN:  CLR     A               ;INITIALIZE THINGS
000A 85                 CLR     F0
000B C5                 SEL     RB0
000C AF                 MOV     STAT,A          ;CLEAR STS IMAGE
000D 90                 MOV     STS,A           ;CLEAR STS
000E D5                 SEL     RB1
000F AB                 MOV     WSTAT,A         ;CLEAR STATUS
0010 8906               ORL     P1,#STP OR SLOW ;STOP DRIVE AND SELECT SLOW FOR STARTERS
0012 99F6               ANL     P1,#FORWD AND RD;SELECT FORWARD AND READ
0014 8A02               ORL     P2,#DAOFF       ;TURN OFF DRIVE ACTIVE LED
0016 F5                 EN      FLAGS           ;ENABLE FLAG INTERRUPT OUTPUTS
                ;
                ;COMMAND RECOGNIZER MAIN LOOP
                ;
0017 C5         B1:     SEL     RB0             ;COMMAND PROCESSING IN RB0
0018 D61F               JNIBF   B2              ;TEST IF IBF INPUT
001A 7623               JF1     CMDIN           ;YES THERE IS AN INPUT, SO TEST IF ITS A COMMAND
001C 22                 IN      A,DBB           ;NOPE, ITS DATA SO IGNORE IT
001D 0417               JMP     B1              ;JUST GO BACK TO TEST IBF
001F 14AD       B2:     CALL    STSUP           ;NO INPUT, UPDATE STS WITH DRIVE STATUS
0021 0417               JMP     B1              ;GO BACK TO TEST IBF
                ;
                ;**********************************************************************
                ;
                ;COMMAND PROCESSOR - TESTS VALIDITY OF INPUT AND BRANCHES TO THE APPROPRIATE
                ;ROUTINE.  ILLEGAL COMMANDS ARE FLAGGED AS COMMAND ERRORS.
                ;
                ;**********************************************************************
                ;
0023 FF         CMDIN:  MOV     A,STAT          ;GET STS IMAGE
0024 4380               ORL     A,#BUSY         ;SET BUSY FOR ALL COMMAND INPUTS
0026 AF                 MOV     STAT,A          ;RESTORE IMAGE
0027 90                 MOV     STS,A           ;UPDATE STS
0028 22                 IN      A,DBB           ;READ COMMAND FROM DBBIN
0029 AA                 MOV     CMDSAV,A        ;SAVE IT IN CMDSAV
002A BC06               MOV     TEMP1,#6H       ;INITIALIZE ILLEGAL COMMAND COUNTER
002C FA         CMDIN1: MOV     A,CMDSAV        ;GET COMMAND FROM CMDSAV
002D 17                 INC     A
002E DC                 XRL     A,TEMP1         ;TEST IF VALID
002F C63A               JZ      CMDIN2          ;YES, INDIRECT JUMP TO IT
0031 EC2C               DJNZ    TEMP1,CMDIN1    ;NO MATCH YET, TRY AGAIN
                                                ;NO MATCH, COMMAND ERROR
                ;
0033 54C4       CMDIN3: CALL    NDRACT          ;RESET DRACT AND BUSY (DRACT WAS NEVER SET)
0035 2302               MOV     A,#CMDERR       ;COMMAND ERROR CODE
0037 02                 OUT     DBB,A           ;OUTPUT ERROR CODE
0038 0417               JMP     B1              ;GO BACK TO TEST FOR IBF
                ;
003A FA         CMDIN2: MOV     A,CMDSAV        ;IT'S A GOOD COMMAND, GET IT FROM CMDSAV
003B 033E               ADD     A,#(LOW CMDJMP) ;ADD OFFSET
003D B3                 JMPP    @A              ;INDIRECT JUMP TO THE COMMAND ROUTINE THRU TABLE;
                ;COMMAND JUMP TABLE
                ;
003E 44         CMDJMP: DB      (LOW RESJMP)
003F 46                 DB      (LOW REDJMP)
0040 48                 DB      (LOW WRTJMP)
0041 4A                 DB      (LOW SKPJMP)
0042 4C                 DB      (LOW REWJMP)
0043 44                 DB      (LOW ARTJMP)
                ;
                ARTJMP:
0044 044E       RESJMP: JMP     RESCOM
0046 2400       REDJMP: JMP     READ
0048 0455       WRTJMP: JMP     WRITE
004A 4414       SKPJMP: JMP     SKIP
004C 4494       REWJMP: JMP     REWND
                ;
                ;IT'S A ABORT COMMAND WHILE IN COMMAND RECOGNIZER LOOP
                ;
004E 54C4       RESCOM: CALL    NDRACT          ;RESET BUSY AND DRACT (DRACT NEVER WAS SET)
0050 2301               MOV     A,#ABTCMP       ;GET ABORT COMPLETE CODE
0052 02                 OUT     DBB,A           ;OUTPUT IT
0053 0400               JMP     RESET           ;GO START OVER
                ;
                $EJECT
```

```
  LOC  OBJ         SOURCE STATEMENT

                ;**********************************************************************
                ;
                ;WRITE TO TAPE ROUTINE
                ;
                ;**********************************************************************
                ;
  0055 C5       WRITE:  SEL     RB0
  0056 85               CLR     F0                ;CLEAR INT COUNT FLAG
  0057 65               STOP    TCNT              ;BE SURE THAT THE TIMER IS STOPPED
  0058 35               DIS     TCNTI             ;DISABLE TIMER INTS
  0059 1659     CLRTF:  JTF     CLRTF             ;BE SURE THAT THE TIMER FLAG IS CLEARED
  005B D65B     WR1:    JNIBF   WR1               ;WAIT FOR BYTE COUNT LSB
  005D 22               IN      A,DBB             ;READ COUNT LSB FROM DBBIN
  005E 7633             JF1     CMDIN3            ;TEST IF COMMAND - ERROR
  0060 A8               MOV     CNTLSB,A          ;IT'S DATA SO STORE IT AWAY
  0061 D661     WR2:    JNIBF   WR2               ;WAIT FOR BYTE COUNT MSB
  0063 22               IN      A,DBB             ;READ COUNT MSB FROM DBBIN
  0064 7633             JF1     CMDIN3            ;TEST IF COMMAND - ERROR
  0066 A9               MOV     CNTMSB,A          ;IT'S DATA SO STORE IT AWAY
  0067 F8               MOV     A,CNTLSB          ;GET COUNT LSB
  0068 17               INC     A                 ;INC IT TO ACCOUNT FOR SYNC
  0069 A8               MOV     CNTLSB,A          ;SAVE IT
  006A 966D             JNZ     NINMSB            ;NO OVERFLOW, DON'T INC COUNT MSB
  006C 19               INC     CNTMSB            ;OVERFLOW, SO INC COUNT MSB
  006D 09       NINMSB: IN      A,P1              ;GET DRIVE STATUS
  006E D2C5             JB6     DRIVER            ;TEST IF NO TAPE
  0070 B2C5             JB5     DRIVER            ;TEST IF FILE PROTECTED
                                                  ;EXIT WITH ERROR IF EITHER
  0072 BB00             MOV     CHKSUM,#00H       ;CLEAR CHECKSUM REGISTER
  0074 D5               SEL     RB1
  0075 BC08             MOV     BITCNT,#WRCNT     ;INITIALIZE WRITE BIT COUNTER
  0077 BDAA             MOV     SERIAL,#SYNC      ;LOAD SYNC INTO SERIAL FOR 1ST BYTE
  0079 23FC             MOV     A,#WRTIM          ;GET WRITE TIMER CONSTANT (1/2 CELL TIME)
  007B 62               MOV     T,A               ;LOAD TIMER BUT DON'T START IT YET
  007C FB               MOV     A,WSTAT           ;GET WRITE STATUS
  007D B2A9             JB5     WEOTER            ;IF EOTFLG SET, STILL AT END OF TAPE - ERROR
  007F C5               SEL     RB0
  0080 54BC             CALL    DRACTS            ;NOT AT EOT SO SET DRIVE ACTIVE AND CONTINUE
  0082 D5               SEL     RB1
  0083 890E             ORL     P1,#WR OR SLOW OR STP   ;SETUP PORT FOR SLOW WRITE
  0085 99FC             ANL     P1,#SRT AND FORWD       ;START DRIVE IN FORWARD
  0087 FB               MOV     A,WSTAT           ;GET WRITE STATUS AGAIN
  0088 37               CPL     A                 ;COMP FOR 0 TEST
  0089 D28D             JB6     WR3               ;TEST BOTFLG - WRITE OVER HOLE IF SET
  008B 54DC             CALL    PASHOL            ;GET OFF CLEAR LEADER AND PAST HOLE IN TAPE
  008D BB80     WR3:    MOV     WSTAT,#80H        ;SETUP WRITE STATUS WITH WRFLG SET
  008F C5               SEL     RB0
  0090 14C1             CALL    DEL150            ;WAIT 450 MS IRG BEFORE WRITING DATA
  0092 14C1             CALL    DEL150
  0094 14C1             CALL    DEL150
  0096 55               STRT    T                 ;START TIMER
  0097 25               EN      TCNTI             ;ENABLE TIMER INTERRUPTS
                ;
                ;TIMER INTERRUPT ROUTINE DOES ALL THE WORK SO WAIT UNTIL IT RESETS WRFLG
                ;
  0098 C5       WR4:    SEL     RB0
  0099 14AD             CALL    STSUP             ;UPDATE STS WHILE WAITING
  009B D5               SEL     RB1
  009C FB               MOV     A,WSTAT           ;GET WRITE STATUS
  009D F298             JB7     WR4               ;TEST IF WRITE DONE (WRFLG RESET)
                ;
                ;WRFLG IS RESET SO WRITE OPERATION MUST BE COMPLETE - OUTPUT RESULT
                ;
  009F C5       WR5:    SEL     RB0
  00A0 54C4             CALL    NDRACT            ;RESET DRACT AND BUSY
  00A2 D5               SEL     RB1
  00A3 FA               MOV     A,RESULT          ;GET RESULT CODE
  00A4 02               OUT     DBB,A             ;OUTPUT IT
  00A5 14C1             CALL    DEL150            ;WAIT FOR DRIVE TO STOP
                                                  ;FULLY BEFORE ACCEPTING NEW COMMAND
  00A7 0417             JMP     B1                ;DONE, RETURN TO COMMAND RECOGNIZER LOOP
                ;
                ;TAPE IS AT EOT WHEN WRITE COMMAND ISSUED - EXIT WITH ERROR
                ;
  00A9 BA83     WEOTER: MOV     RESULT,#EOTERR    ;EOT ERROR RESULT CODE
  00AB 049F             JMP     WR5               ;GO RESET BUSY AND OUTPUT RESULT
                ;
                $EJECT
```

```
LOC  OBJ        LINE        SOURCE STATEMENT

                ;**********************************************************************
                ;
                ;STS UPDATE SUBROUTINE - UPDATES THE CASSETTE PRESENCE AND FILE PROTECT
                ;BIT IN STS.  (ENTER AND EXIT IN RBO)
                ;
                ;**********************************************************************
                ;
00AD FF         STSUP:  MOV     A,STAT             ;GET STS IMAGE
00AE 4360               ORL     A,#TAPIN OR FILPRT        ;SET BOTH PRESENCE AND FILE PROTECT
00B0 AF                 MOV     STAT,A             ;RESTORE IMAGE
00B1 09                 IN      A,P1               ;READ INPUT
00B2 439F               ORL     A,#NOT(TAPIN OR FILPRT) ;SET BITS TO CORRECT STATE
00B4 5F                 ANL     A,STAT
00B5 AF                 MOV     STAT,A             ;RESTORE IMAGE
00B6 90                 MOV     STS,A              ;UPDATE STS
00B7 83                 RET
                ;
                ;**********************************************************************
                ;
                ;DELAY ROUTINES- ENTER/EXIT IN RBO
                ;
                ;**********************************************************************
                ;
00B8 BD24       DEL50:  MOV     R5,#36D            ;50MS DELAY ROUTINE
00BA BEFF       DEL1:   MOV     R6,#0FFH
00BC EEBC       DEL2:   DJNZ    R6,DEL2
00BE EDBA               DJNZ    R5,DEL1
00C0 83                 RET
                ;
00C1 BD6C       DEL150: MOV     R5,#108D           ;150MS DELAY ROUTINE
00C3 04BA               JMP     DEL1
                ;
                ;**********************************************************************
                ;
                ;DRIVE ERROR EXIT - NO TAPE OR FILE IS PROTECTED FOR WRITE
                ;
                ;**********************************************************************
                ;
00C5 C5         DRIVER: SEL     RBO
00C6 54C4               CALL    NDRACT             ;RESET DRACT AND BUSY
00C8 09                 IN      A,P1               ;READ DRIVE STATUS
00C9 D2D0               JB6     NT                 ;TEST IF TAPE IS THERE
00CB 2304               MOV     A,#NWR             ;TAPE IS THERE SO ERROR MUST BE FILE PROTECT
00CD 02         DR1:    OUT     DBB,A              ;OUTPUT ERROR CODE
00CE 0417               JMP     B1                 ;RETURN TO COMMAND LOOP
00D0 2303       NT:     MOV     A,#NTAPE           ;NO TAPE ERROR
00D2 04CD               JMP     DR1
                ;
                ;**********************************************************************
                ;
                ;READ ERROR WITH ADVANCE TO IRG BEFORE STOPPING DRIVE.
                ;WAIT FOR OBF TO BE FREE BEFORE RESETTING BUSY THEN OUTPUT RESULT.
                ;RDERR3 LABEL IS EXIT POINT FOR OTHER ROUTINES NEEDING TO WAIT FOR
                ;OBF TO BE FREE BEFORE OUTPUTTING RESULT.
                ;ROUTINE EXITS IN RBO.
                ;
                ;**********************************************************************
                ;
00D4 C5         RDERR:  SEL     RBO
00D5 BC01               MOV     BLKCNT,#1H         ;SET SKIP COUNTER TO ADVANCE TO NEXT IRG
00D7 BD33               MOV     BLKTIM,#SLWIRG     ;SETUP IRG COUNTER
00D9 BE33               MOV     BLKSAV,#SLWIRG
00DB 5400               CALL    SKIPER             ;DO SKIP TO NEXT IRG
00DD F6F1               JC      RDERR2             ;TEST IF EOT FOUND
00DF 8902       RDERR3: ORL     P1,#STP            ;STOP DRIVE WHEN DONE
00E1 8A02               ORL     P2,#DAOFF          ;TURN OFF DRIVE ACTIVE LED
00E3 D6E7       RDERR4: JNIBF   RDERR5             ;TEST IBF WHILE WAITING FOR OBF
00E5 04F8               JMP     RDERR6             ;IBF SET - GO TEST INPUT
00E7 86E3       RDERR5: JOBF    RDERR4             ;TEST OBF, LOOP IF 1, CONTINUE IF 0
00E9 C5                 SEL     RBO
00EA 54C4               CALL    NDRACT             ;RESET DRACT AND BUSY
00EC D5                 SEL     RB1
00ED FA                 MOV     A,RESULT           ;GET RESULT
00EE 02                 OUT     DBB,A              ;OUTPUT RESULT
00EF 0417               JMP     B1                 ;GO BACK TO COMMAND LOOP
                ;
00F1 D5         RDERR2: SEL     RB1                ;EOT FOUND WHILE SKIPPING
00F2 BA46               MOV     RESULT,#REOTER     ;RESET RESULT VALUE TO EOT ERROR
00F4 BB20               MOV     RSTAT,#EOTFLG      ;SET EOTFLG IN RSTAT
00F6 04DF               JMP     RDERR3             ;GO OUTPUT NEW RESULT
                ;
00F8 22         RDERR6: IN      A,DBB              ;READ INPUT
00F9 D305               XRL     A,#ABORT           ;TEST IF ABORT
00FB 96E7               JNZ     RDERR5             ;IGNORE IT IF NOT
00FD 044E               JMP     RESCOM             ;IT'S AN ABORT, GO RESET
                ;
                $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

  LOC  OBJ         LINE        SOURCE STATEMENT

  0100             514          ORG     100H
                   515 ;
                   516 ;*********************************************************************
                   517 ;
                   518 ;READ FROM TAPE ROUTINE
                   519 ;
                   520 ;*********************************************************************
                   521 ;
  0100 C5          522 READ:    SEL     RB0
  0101 65          523          STOP    TCNT             ;BE SURE THE TIMER IS STOPPED
  0102 35          524          DIS     TCNTI            ;DISABLE TIMER INTS
  0103 1603        525 RCLRTF:  JTF     RCLRTF           ;BE SURE THE TIMER FLAG IS CLEARED
  0105 85          526          CLR     F0               ;CLEAR LAST DATA FLAG
  0106 2320        527          MOV     A,#20H           ;GET POINTER START LOCATION
  0108 A8          528          MOV     LBOUT,A          ;INITIALIZE LBOUT
  0109 A9          529          MOV     LBRDY,A          ;INITIALIZE LBRDY
  010A BB00        530          MOV     CHKSUM,#00H      ;CLEAR CHECKSUM LOCATION
  010C D5          531          SEL     RB1
  010D A8          532          MOV     LBIN,A           ;INITIALIZE LBIN
  010E BC08        533          MOV     BITCNT,#RDCNT    ;INITIALIZE READ BIT COUNTER
  0110 09          534          IN      A,P1             ;READ DRIVE STATUS
  0111 D24F        535          JB6     DRIVJ            ;TEST IF TAPE IS THERE TO READ
  0113 C5          536          SEL     RB0
  0114 54BC        537          CALL    DRACTS           ;SET DRIVE ACTIVE
  0116 23FA        538          MOV     A,#RDTIM.        ;GET READ TIMER CONSTANT (3/4 CELL TIME)
  0118 62          539          MOV     T,A              ;LOAD TIMER BUT DON'T START IT YET
  0119 25          540          EN      TCNTI            ;ENABLE TIMER INTERRUPTS
  011A D5          541          SEL     RB1
  011B FB          542          MOV     A,RSTAT          ;GET READ STATUS
  011C B24B        543          JB5     REOT             ;TEST IF AT EOT - ERROR IF SO
  011E 8904        544          ORL     P1,#SLOW         ;SELECT SLOW
   120 99F4        545          ANL     P1,#RD AND FORWD AND SRT       ;START DRIVE, FORWARD AND READ
  0122 37          546          CPL     A                ;COMP A FOR 0 TEST (STILL HAVE RSTAT)
  0123 D227        547          JB6     RD1              ;TEST FOR AT BOT, IF NOT JUST LOOK FOR MARK
  0125 54DC        548          CALL    PASHOL           ;IF BOT, WAIT UNITL PAST CLEAR LEADER AND HOLE
  0127 BB06        549 RD1:     MOV     RSTAT,#06H       ;SETUP READ STATUS - SNBFLG AND STRFLG SET
  0129 14C1        550          CALL    DEL150           ;LET DRIVE START UP
                   551                                   ;AND WAIT OVER WRITE STOP LOCATION
  012B 462B        552 RD1A:    JNT1    RD1A             ;WAIT FOR MARK
  012D 562D        553 RD2:     JT1     RD2              ;WAIT FOR TRANSITION TO SPACE
  012F 55          554          STRT    T                ;START TIMER
                   555 ;
                   556 ;LOOP START - LOOK FOR READ STATUS FLAGS BEING SET BY TIMER INTERRUPT ROUTINE
                   557 ;
  0130 D5          558 RD3:     SEL     RB1
  0131 D635        559          JNIBF   RD4              ;TEST FOR IBF EVEN WHEN READING
  0133 24D5        560          JMP     RDIBF            ;INPUT DURING READ - GO TEST IT
  0135 FB          561 RD4:     MOV     A,RSTAT          ;GET READ STATUS
  0136 1251        562          JB0     GETDAT           ;TEST DATA READY FLAG (RDYFLG)
  0138 7291        563          JB3     IRGFND           ;TEST IRG FLAG (IRGFLG)
  013A B24B        564          JB5     REOT             ;EOT FOUND DURING READ (EOTFLG SET) - ERROR
                   565 ;
                   566 ;NOTHING FROM TIMER INTERRUPT ROUTINE SO GO HANDLE CIRCULAR BUFFER.
                   567 ;
  013C C5          568          SEL     RB0
  013D F9          569          MOV     A,LBRDY          ;GET READY POINTER
  013E D8          570          XRL     A,LBOUT          ;COMPARE TO OUT POINTER
  013F C630        571          JZ      RD3              ;EMPTY IF THE SAME SO JUST LOOP
                   572                                   ;NOT EMPTY SO SEE IF NEXT BYTE CAN BE OUTPUT
  0141 8630        573          JOBF    RD3              ;TEST DBBOUT - FULL, LOOP
  0143 F0          574          MOV     A,@LBOUT         ;DBBOUT FREE - GET DATA
  0144 02          575          OUT     DBB,A            ;OUTPUT IT
  0145 F8          576          MOV     A,LBOUT          ;GET OUT POINTER
  0146 54CC        577          CALL    BUMPIT           ;BUMP POINTER
  0148 A8          578          MOV     LBOUT,A          ;RETURN IT
  0149 2430        579          JMP     RD3              ;LOOP
                   580 ;
                   581 ;TAPE AT EOT WHEN READ COMMAND ISSUED - ERROR
                   582 ;
  014B BA46        583 REOT:    MOV     RESULT,#REOTER   ;EOT AT READ ERROR CODE
  014D 04DF        584          JMP     RDERR3           ;EXIT
                   585 ;
                   586 ;OUT OF PAGE JUMP FOR DRIVE ERROR
                   587 ;
  014F 04C5        588 DRIVJ:   JMP     DRIVER
                   589 ;
                   590 ;TIMER ROUTINE FLAGGED DATA IS READY
                   591 ;
  0151 53FE        592 GETDAT:  ANL     A,#NOT RDYFLG    ;RESET DATA READY FLAG (RDYFLG)
  0153 AB          593          MOV     RSTAT,A          ;RESTORE READ STATUS
  0154 3282        594          JB1     SNBTST           ;TEST IF DATA SHOULD BE SYNC (SNBFLG SET)
  0156 C5          595          SEL     RB0              ;NO, TRY TO PUT IN BUFFER
  0157 F8          596          MOV     A,LBOUT          ;GET OUT POINTER
  0158 D5          597          SEL     RB1
  0159 54D3        598          CALL    DUMPIT           ;DUMP IT FOR FULL TEST
  015B D8          599          XRL     A,LBIN           ;COMPARE IT TO IN POINTER
  015C 9662        600          JNZ     NOFULL           ;IF NOT SAME, THEN BUFFER ISN'T FULL
  015E BA41        601          MOV     RESULT,#OVERUN   ;BUFFER IS FULL SO OVERRUN ERROR CODE
  0160 04D4        602          JMP     RDERR            ;GO EXIT FROM ERROR, SKIP TO NEXT IRG
  0162 FE          603 NOFULL:  MOV     A,RDATA          ;BUFFER ISN'T FULL SO GET DATA FROM HOLDING
  0163 C5          604          SEL     RB0
  0164 6B          605          ADD     A,CHKSUM         ;ADD IT TO CHECKSUM
  0165 AB          606          MOV     CHKSUM,A         ;RESTORE CHECKSUM
  0166 D5          607          SEL     RB1
```

```
 LOC  OBJ         LINE        SOURCE STATEMENT

0167 FE                    MOV     A,RDATA          ;GET DATA AGAIN
0168 A0                    MOV     @LBIN,A          ;PUT IT IN BUFFER
0169 F8                    MOV     A,LBIN           ;GET IN POINTER
016A 54CC                  CALL    BUMPIT           ;BUMP IT
016C A8                    MOV     LBIN,A           ;RETURN IT
016D FB                    MOV     A,RSTAT          ;GET READ STATUS
016E 5277                  JB2     LBTST            ;TEST IF LBRDY SHOULD BE
                                                    ;BUMPED TOO (SRTFLG RESET)
0170 C5                    SEL     RB0
0171 F9                    MOV     A,LBRDY          ;SRTFLG IS RESET SO GET LBRDY
0172 54CC                  CALL    BUMPIT           ;BUMP IT
0174 A9                    MOV     LBRDY,A          ;RETURN IT
0175 2430                  JMP     RD3              ;GO BACK TO LOOK FOR DATA
                 ;
                 ;START FLAG IS SET - SEE IF LBIN HAS ADVANCED FAR ENOUGH TO START INC LBRDY
                 ;
0177 F8          LBTST:    MOV     A,LBIN           ;GET IN POINTER
0178 D322                  XRL     A,#22H           ;TEST IF READY POINTER SHOULD BE BUMPED
017A 9630                  JNZ     RD3              ;NO, GO BACK FOR DATA
017C FB                    MOV     A,RSTAT          ;YES, GET READ STATUS
017D 53FB                  ANL     A,#NOT STRFLG    ;RESET START FLAG
017F AB                    MOV     RSTAT,A          ;RESTORE STATUS
0180 2430                  JMP     RD3              ;GO BACK TO LOOK FOR DATA
                 ;
                 ;DATA SHOULD BE SYNC - TEST IT
                 ;
0182 FE          SNBTST:   MOV     A,RDATA          ;GET DATA
0183 D3AA                  XRL     A,#SYNC          ;COMPARE TO SYNC
0185 C68B                  JZ      RSNB             ;IT'S A SYNC, RESET SNBFLG
0187 BA42                  MOV     RESULT,#SYNC1    ;IT'S NOT A SYNC, BAD FIRST SYNC ERROR CODE
0189 04D4                  JMP     RDERR            ;EXIT READ ERROR, ADVANCE TO NEXT IRG
018B FB          RSNB:     MOV     A,RSTAT          ;SYNC TEST IS OK, GET READ STATUS
018C 53FD                  ANL     A,#NOT SNBFLG    ;RESET SNB FLAG
018E AB                    MOV     RSTAT,A          ;RESTORE STATUS
018F 2430                  JMP     RD3              ;GO BACK TO LOOK FOR DATA
                 ;
                 ;IRG FOUND - TEST FOR SYNC, CORRECT AND TEST CHECKSUM
                 ;
0191 C5          IRGFND:   SEL     RB0
0192 FF                    MOV     A,STAT           ;GET STS IMAGE
0193 53EF                  ANL     A,#NOT DRACT     ;RESET DRIVE ACTIVE
0195 AF                    MOV     STAT,A           ;RESTORE IMAGE
0196 90                    MOV     STS,A            ;UPDATE STS
0197 8A02                  ORL     P2,#DAOFF        ;TURN OFF DRIVE ACTIVE-LED
0199 D5                    SEL     RB1
019A F8                    MOV     A,LBIN           ;GET IN POINTER
019B 54D3                  CALL    DUMPIT           ;DEC IT TO POINT AT LAST DATA, ALIAS SYNC
019D A8                    MOV     LBIN,A           ;RETURN IT
019E F0                    MOV     A,@LBIN          ;GET LAST DATA
019F D3AA                  XRL     A,#SYNC          ;COMPARE TO SYNC
01A1 96BA                  JNZ     IRGF1            ;NOT EQUAL - ERROR
01A3 F8                    MOV     A,LBIN           ;GET POINTER AGAIN
01A4 54D3                  CALL    DUMPIT           ;DEC IT TO POINT AT 2ND
                                                    ;TO LAST DATA, ALIAS CHECKSUM
01A6 A8                    MOV     LBIN,A           ;RETURN IT
01A7 C5                    SEL     RB0
01A8 FB                    MOV     A,CHKSUM         ;GET ACCUMULATED CHECKSUM
01A9 0356                  ADD     A,#56H           ;SUBTRACT OUT SYNC
01AB AB                    MOV     CHKSUM,A         ;RESTORE CHECKSUM
01AC D5                    SEL     RB1
01AD F0                    MOV     A,@LBIN          ;GET RECEIVED CHECKSUM
01AE C5                    SEL     RB0
01AF 37                    CPL     A                ;SUBTRACT IT OUT - MAKE IT MINUS
01B0 17                    INC     A                ;
01B1 6B                    ADD     A,CHKSUM         ;SUBTRACT FROM ACC CHECKSUM
01B2 D5                    SEL     RB1
01B3 D0                    XRL     A,@LBIN          ;COMPARE RESULT TO RECEIVED
01B4 96BE                  JNZ     CKSER            ;NOT EQUAL, THEN CHECKSUM ERROR
01B6 BA00                  MOV     RESULT,#GOOD     ;EQUAL, THEN GOOD RESULT
01B8 24C0                  JMP     BUFFER           ;GO FINISH OFF BUFFER BEFORE OUTPUTTING RESULT
01BA BA43        IRGF1:    MOV     RESULT,#SYNC2    ;2ND SYNC ERROR CODE
01BC 04DF                  JMP     RDERR3           ;EXIT
01BE BA44        CKSER:    MOV     RESULT,#BADCHS   ;BAD CHECKSUM ERROR CODE BUT STILL FINISH BUFFER
                 ;
                 ;DONE WITH READ - LET BUFFER EMPTY BEFORE OUTPUTTING RESULT
                 ;
01C0 C5          BUFFER:   SEL     RB0
01C1 F8                    MOV     A,LBOUT          ;GET OUT POINTER
01C2 D9                    XRL     A,LBRDY          ;COMPARE TO READY POINTER
01C3 96C7                  JNZ     BUF1             ;NOT EMPTY YET SO GO TEST OBF
01C5 04DF                  JMP     RDERR3           ;BUFFER IS EMPTY - GO OUTPUT RESULT
01C7 D6CB        BUF1:     JNIBF   BUF2             ;TEST FOR INPUT
01C9 24D5                  JMP     RDIBF            ;IF INPUT, GO TEST IT
01CB 86C7        BUF2:     JOBF    BUF1             ;TEST OBF
01CD F0                    MOV     A,@LBOUT         ;OBF FREE, GET DATA FROM BUFFER
01CE 02                    OUT     DBB,A            ;OUTPUT IT
01CF F8                    MOV     A,LBOUT          ;GET OUT POINTER
01D0 54CC                  CALL    BUMPIT           ;BUMP IT TO POINT AT NEXT DATA
01D2 A8                    MOV     LBOUT,A          ;RETURN IT
01D3 24C0                  JMP     BUFFER           ;GO TEST IT DONE
                 ;
                 ;IBF FOUND DURING READ OPERATION - TEST IF ABORT, IGNORE IF NOT
                 ;
 LOC  OBJ             SOURCE STATEMENT

01D5 22          RDIBF:   IN      A,DBB              ;READ DBBIN
01D6 76DA                 JF1     ABTST              ;TEST FOR COMMAND
01D8 2430                 JMP     RD3                ;MUST BE DATA, IGNORE IT
01DA D305        ABTST:   XRL     A,#ABORT           ;COMPARE TO ABORT COMMAND
01DC 9630                 JNZ     RD3                ;NOT EQUAL, IGNORE IT
01DE BA01        ABTST1:  MOV     RESULT,#ABTCMP     ;IT IS AN ABORT, ABORT COMPLETE RESULT CODE
01E0 04D4                 JMP     RDERR              ;EXIT LIKE IT WAS AN ERROR, ADVANCE TO IRG

                 $EJECT

0200                      ORG     200H
                 ;
                 ;*******************************************************************
                 ;
                 ;SKIPER SUBROUTINE - ADVANCES TO NEXT IRG BASED ON DIRECTION AND
                 ;SPEED PASSED IN BLKTIM.
                 ;CARRY=0, NO EOT ENCOUNTERED.  CARRY=1, EOT ENCOUNTERED
                 ;ENTER AND EXIT IN RB0
                 ;
                 ;*******************************************************************
                 ;
0200 97          SKIPER:  CLR     C                  ;CLEAR EOT INTERNAL FLAG
0201 09                   IN      A,P1               ;READ DRIVE STATUS
0202 9212                 JB4     SKIPR3             ;TEST FOR CLEAR LEADER
0204 4600                 JNT1    SKIPER             ;NO CLEAR LEADER, WAIT UNTIL INPUT IS HIGH
0206 560C        SKIPR1:  JT1     SKIPR2             ;WHILE INPUT IS HIGH, DEC BLKTIM COUNTER
0208 FE                   MOV     A,BLKSAV           ;INPUT WENT LOW, RESET BLKTIM COUNTER
0209 AD                   MOV     BLKTIM,A
020A 4400                 JMP     SKIPER             ;GO WAIT UNTIL INPUT IS HIGH AGAIN
020C 09          SKIPR2:  IN      A,P1               ;READ DRIVE STATUS
020D 9212                 JB4     SKIPR3             ;TEST CLEAR LEADER - ERROR IF TRUE
020F ED06                 DJNZ    BLKTIM,SKIPR1      ;INPUT STILL HIGH, DEC BLKTIM COUNTER
0211 83                   RET                        ;RETURN WHEN AT IRG
0212 A7          SKIPR3:  CPL     C                  ;SET CARRY TO SHOW EOT
0213 83                   RET                        ;RETURN
                 ;
                 ;*******************************************************************
                 ;
                 ;SKIP COMMAND ROUTINE - NEXT DATA BYTE IS NUMBER OF IRG'S TO SKIP
                 ;
                 ;*******************************************************************
                 ;
0214 D614        SKIP:    JNIBF   SKIP               ;WAIT FOR SKIP COUNT INPUT
0216 7692                 JF1     CMDINJ             ;TEST IF COMMAND INSTEAD - EXIT IF YES
0218 85                   CLR     F0                 ;CLEAR DIRECTION FLAG - DEFAULT FORWARD
0219 54BC                 CALL    DRACTS             ;GO SET DRIVE ACTIVE
021B 8904                 ORL     P1,#SLOW           ;START OUT SLOW
021D 22                   IN      A,DBB              ;READ SKIP COUNT INPUT
021E AC                   MOV     BLKCNT,A           ;SAVE IT IN BLOCK COUNTER
021F F262                 JB7     RSKIP              ;IF BIT 7 SET, IT'S A REVERSE SKIP
                 ;
                 ;FORWARD SKIP
                 ;
0221 D5                   SEL     RB1
0222 FB                   MOV     A,RSTAT            ;GET READ STATUS
0223 B278                 JB5     SKIP8              ;STATUS SAYS WE'RE AT EOT - EXIT WITH ERROR
0225 99F4                 ANL     P1,#FORWD AND SRT AND RD       ;IT'S GO - FORWARD
0227 C5                   SEL     RB0
0228 37                   CPL     A                  ;COMP A FOR 0 TEST
0229 D22D                 JB6     SKIP2              ;WE'RE NOT AT BOT SO JUST DO SKIP
022B 54DC        SKIP1:   CALL    PASHOL             ;AT BOT SO GET PAST CLEAR LEADER AND HOLE
022D 14C1        SKIP2:   CALL    DEL150             ;WAIT OUT JUNK AT BEGINNING OF EACH BLOCK
022F B644        SKIP3:   JF0     SKIP6              ;DON'T WORRY ABOUT FAST OR SLOW WHEN REVERSE
0231 FC                   MOV     A,BLKCNT           ;GET BLOCK COUNT
0232 03F8                 ADD     A,#-8H             ;SEE IF COUNT IS >8
0234 F63E                 JC      SKIP4              ;YES, USE FAST IRG TIMING
0236 BD33                 MOV     BLKTIM,#SLWIRG     ;COUNT IS <8, USE SLOW IRG TIMING
0238 BE33                 MOV     BLKSAV,#SLWIRG
023A 8904                 ORL     P1,#SLOW           ;SELECT SLOW
023C 4444                 JMP     SKIP6              ;GO DO SKIP
023E BD20        SKIP4:   MOV     BLKTIM,#FASIRG     ;COUNT IS >8, USE FAST IRG TIMING
0240 BE20                 MOV     BLKSAV,#FASIRG
0242 99FB                 ANL     P1,#FAST           ;SELECT FAST
0244 464B        SKIP6:   JNT1    SKIP7              ;WAIT FOR SPACE TO START IRG FIND
0246 09                   IN      A,P1               ;READ DRIVE STATUS
0247 9278                 JB4     SKIP8              ;TEST CLEAR LEADER - EXIT IF FOUND
0249 4444                 JMP     SKIP6              ;CONTINUE TO WAIT FOR SPACE
024B D64F        SKIP7:   JNIBF   SKIP12             ;TEST IBF WHILE SKIPPING
024D 4487                 JMP     SKIP11             ;IBF SET, GO TEST IT
024F 5400        SKIP12:  CALL    SKIPER             ;DO SKIP TO IRG
0251 F678                 JC      SKIP8              ;TEST IF EOT OR BOT FOUND
0253 EC2D                 DJNZ    BLKCNT,SKIP2       ;DO IT FOR ALL BLOCK COUNT
0255 B65E                 JF0     SKIP13             ;DELAY A LITTLE IF REVERSE
0257 D5          SKIP14:  SEL     RB1
0258 BA00                 MOV     RESULT,#GOOD       ;GOOD RESULT
025A BB00                 MOV     RSTAT,#00H         ;CLEAR READ STATUS
025C 04DF                 JMP     RDERR3             ;USE READ EXIT TO COMPLETE
                 ;
                 ;REVERSE SKIP DELAY WHEN BLOCK COUNT EXPIRED
                 ;
025E 14B8        SKIP13:  CALL    DEL50
0260 4457                 JMP     SKIP14
                 ;
 LOC  OBJ          LINE        SOURCE STATEMENT

                   ;REVERSE SKIP IS DESIRED - SETUP DRIVE AND DIRECTION FLAG
                   ;
0262 95            RSKIP:  CPL     F0                  ;SET DIRECTION FLAG
0263 537F                  ANL     A,#7FH              ;MASK OFF DIRECTION
0265 AC                    MOV     BLKCNT,A            ;RESTORE BLKCNT
0266 BD20                  MOV     BLKTIM,#RWDIRG      ;SET REWIND BLOCK TIMER
0268 BE20                  MOV     BLKSAV,#RWDIRG
026A D5                    SEL     RB1
026B FB                    MOV     A,RSTAT             ;GET READ STATUS
026C D278                  JB6     SKIP8               ;AT BOT SO EXIT WITH ERROR
026E 8901                  ORL     P1,#REWIND          ;SELECT REVERSE
0270 99F5                  ANL     P1,#SRT AND RD      ;START DRIVE
0272 C5                    SEL     RB0
0273 37                    CPL     A                   ;COMP A FOR 0 TEST
0274 B22D                  JB5     SKIP2               ;NOT AT EOT SO JUST DO SKIP
0276 442B                  JMP     SKIP1               ;AT EOT SO WAIT PAST CLEAR LEADER AND HOLE
                   ;
                   ;CLEAR LEADER FOUND DURING SKIP OR TAPE ALREADY AT EOT OR BOT
                   ;
0278 D5            SKIP8:  SEL     RB1
0279 B681                  JF0     SKIP9               ;TEST DIRECTION
027B BA47                  MOV     RESULT,#SKPEOT      ;IT'S FORWARD SO IT'S EOT
027D BB20                  MOV     RSTAT,#EOTFLG       ;SET EOT FLAG
027F 04DF                  JMP     RDERR3              ;GO EXIT
0281 BA48          SKIP9:  MOV     RESULT,#SKPBOT      ;IT'S REVERSE SO IT'S BOT
0283 BB40                  MOV     RSTAT,#BOTFLG       ;SET BOT FLAG
0285 04DF                  JMP     RDERR3              ;GO EXIT
                   ;
                   ;IBF FOUND SET DURING SKIP - TEST INPUT
                   ;
0287 22            SKIP11: IN      A,DBB               ;READ INPUT
0288 D305                  XRL     A,#ABORT            ;TEST IF ABORT
028A 964F                  JNZ     SKIP12              ;IGNORE IT IF NOT
028C 8902                  ORL     P1,#STP             ;STOP DRIVE
028E 8A02                  ORL     P2,#DAOFF           ;TURN OFF DRIVE ACTIVE LED
0290 24DE                  JMP     ABTST1              ;YES - EXIT WITH RESULT
                   ;
                   ;OUT OF PAGE JUMP FOR CMDIN
                   ;
0292 0433          CMDINJ: JMP     CMDIN3
                   ;
                   ;*************************************************************************
                   ;
                   ;REWIND COMMAND - STOP WHEN CLEAR LEADER IS FOUND FOR >50MS.
                   ;
                   ;*************************************************************************
                   ;
0294 D5            REWND:  SEL     RB1
0295 FB                    MOV     A,RSTAT             ;GET READ STATUS
0296 D2B8                  JB6     REWND4              ;TEST IF ALREADY AT BOT - EXIT IF YES
0298 C5                    SEL     RB0
0299 54BC                  CALL    DRACTS              ;SET DRIVE ACTIVE
029B 99F3                  ANL     P1,#RD AND FAST     ;SELECT RD AND FAST
029D 8901                  ORL     P1,#REWIND          ;SELECT REWIND
029F 99FD                  ANL     P1,#SRT             ;START DRIVE
02A1 14B8          REWND1: CALL    DEL50               ;WAIT 50MS
02A3 09                    IN      A,P1                ;READ DRIVE STATUS
02A4 92A8                  JB4     REWND2              ;TEST CLEAR LEADER
02A6 44A1                  JMP     REWND1              ;NO CLEAR LEADER - WAIT
02A8 14B8          REWND2: CALL    DEL50               ;WAIT 50MS AGAIN
02AA 09                    IN      A,P1                ;READ DRIVE STATUS AGAIN
02AB 92AF                  JB4     REWND3              ;AT END IF CLEAR LEADER STILL SET
02AD 44A1                  JMP     REWND1              ;OTHERWISE IT WAS JUST HOLE
02AF 8906          REWND3: ORL     P1,#STP OR SLOW     ;STOP DRIVE, SELECT SLOW
02B1 99FE                  ANL     P1,#FORWD           ;SELECT FORWARD
02B3 14B8                  CALL    DEL50               ;WAIT 50MS FOR DRIVE RESET
02B5 D5                    SEL     RB1
02B6 BB40                  MOV     RSTAT,#BOTFLG AND (NOT EOTFLG)  ;SET UP READ STATUS
02B8 BA00          REWND4: MOV     RESULT,#GOOD        ;GOOD RESULT
02BA 04DF                  JMP     RDERR3              ;GO OUTPUT RESULT
                   ;
                   $EJECT
```

```
LOC  OBJ         LINE        SOURCE STATEMENT

                 ;**********************************************************************
                 ;
                 ;DRIVE ACTIVE STATUS SUBROUTINE - ENTER/EXIT IN RB0
                 ;DRIVE ACTIVE BIT IN STATUS IS SET AND DRIVE ACTIVE LED IS TURNED ON
                 ;
                 ;**********************************************************************
                 ;
02BC FF          DRACTS: MOV     A,STAT           ;GET STS IMAGE
02BD 4310                ORL     A,#DRACT         ;SET DRIVE ACTIVE BIT
02BF AF                  MOV     STAT,A           ;RESTORE IMAGE
02C0 90                  MOV     STS,A            ;UPDATE STS
02C1 9AFD                ANL     P2,#DAON         ;TURN ON DRIVE ACTIVE LED
02C3 83                  RET                      ;RETURN
                 ;
                 ;**********************************************************************
                 ;
                 ;DRIVE INACTIVE STATUS SUBROUTINE - ENTER/EXIT IN RB0
                 ;BOTH DRIVE ACTIVE AND BUSY BITS IN STATUS ARE RESET, DRIVE ACTIVE LED IS OFF
                 ;
                 ;**********************************************************************
                 ;
02C4 FF          NDRACT: MOV     A,STAT           ;GET STS IMAGE
02C5 536F                ANL     A,#NOT (BUSY OR DRACT)  ;RESET DRACT AND BUSY
02C7 AF                  MOV     STAT,A           ;RESTORE IMAGE
02C8 90                  MOV     STS,A            ;UPDATE STS
02C9 8A02                ORL     P2,#DAOFF        ;TURN OFF DRIVE ACTIVE LED
02CB 83                  RET
                 ;
                 ;**********************************************************************
                 ;
                 ;BUMPIT - POINTER MANAGEMENT - VALUE IN A IS INCREMENTED AND TESTED
                 ;FOR OVERFLOW.  IF OVERFLOW OCCURS, SET A TO BOTTOM OF BUFFER.
                 ;
                 ;**********************************************************************
                 ;
02CC 17          BUMPIT: INC     A                ;INC A
02CD D2D0                JB6     OVFLOW           ;TEST FOR OVERFLOW
02CF 83                  RET                      ;NO OVERFLOW, RET
02D0 2320        OVFLOW: MOV     A,#20H           ;OVERFLOW SO RESET A
02D2 83                  RET                      ;RETURN
                 ;
                 $EJECT
                 ;**********************************************************************
                 ;
                 ;DUMPIT - POINTER MANAGEMENT - VALUE IN A IS DECREMENTED AND TESTED
                 ;FOR UNDERFLOW.  IF UNDERFLOW OCCURS, SET A TO TOP OF BUFFER.
                 ;
                 ;**********************************************************************
                 ;
02D3 07          DUMPIT: DEC     A                ;DEC A
02D4 37                  CPL     A
02D5 B2D9                JB5     UNFLOW           ;TEST IF UNDERFLOW
02D7 37                  CPL     A                ;NO, COMP BACK
02D8 83                  RET                      ;RETURN
02D9 233F        UNFLOW: MOV     A,#3FH           ;UNDERFLOW SO RESET A
02DB 83                  RET                      ;RETURN
                 ;
                 ;**********************************************************************
                 ;
                 ;SUBROUTINE TO GET PAST HOLE IN TAPE
                 ;
                 ;**********************************************************************
                 ;
02DC 09          PASHOL: IN      A,P1             ;READ DRIVE STATUS
02DD 92DC                JB4     PASHOL           ;WAIT UNTIL OFF CLEAR LEADER
02DF 09          PAS1:   IN      A,P1             ;READ DRIVE STATUS AGAIN
02E0 37                  CPL     A                ;COMP A FOR 0 TEST
02E1 92DF                JB4     PAS1             ;WAIT UNTIL HOLE
02E3 09          PAS2:   IN      A,P1             ;READ DRIVE STATUS
02E4 92E3                JB4     PAS2             ;WAIT UNTIL PAST HOLE
02E6 83                  RET                      ;RETURN
                 ;
                 $EJECT
```

```
 LOC  OBJ         SOURCE STATEMENT

0300                      ORG     300H
                 ;
                 ;***********************************************************************
                 ;
                 ;TIMER INTERRUPT ROUTINES - FIRST DECIDE IF IT'S READ OR WRITE
                 ;
                 ;***********************************************************************
                 ;
0300 D5          INT:     SEL     RB1
0301 AF                   MOV     ASAVE,A            ;SAVE ACCUMULATOR
0302 FB                   MOV     A,RSTAT            ;GET CURRENT STATUS REGISTER
0303 F25E                 JB7     WRINT              ;IF RD/WR FLAG SET, IT'S A WRITE
                                                     ;OTHERWISE, IT'S A READ
                 ;
                 ;***********************************************************************
                 ;
                 ;READ INTERRUPT ROUTINE
                 ;
                 ;***********************************************************************
                 ;
0305 97          RDINT:   CLR     C                  ;CLEAR SHIFTER
0306 560C                 JT1     RDI1               ;TEST INPUT
0308 B612                 JFO     RDI2               ;INPUT=0, TEST LAST
030A 6416                 JMP     SHIFIN             ;INPUT=0, LAST=0, SHIFT IN 0
030C B616        RDI1:    JFO     RDI3               ;INPUT=1, TEST LAST
030E A7                   CPL     C                  ;INPUT=1, LAST=0, SHIFT IN 1
030F 95                   CPL     FO                 ;SET FO TO CURRENT VALUE OF DATA IN
0310 6416                 JMP     SHIFIN
0312 A7          RDI2:    CPL     C                  ;INPUT=0, LAST=1, SHIFT IN 1
0313 95                   CPL     FO                 ;SET FO TO CURRENT VALUE OF DATA IN
0314 6416                 JMP     SHIFIN
                 RDI3:                               ;INPUT=1, LAST=1, SHIFT IN 0
0316 FD          SHIFIN:  MOV     A,DESERL           ;GET CURRENT VALUE OF DATA BYTE
0317 67                   RRC     A                  ;SHIFT IN NEW BIT
0318 AD                   MOV     DESERL,A           ;RESTORE DESERIALIZER
0319 EC22                 DJNZ    BITCNT,RDI4        ;TEST IF BYTE DONE
031B AE                   MOV     RDATA,A            ;IT'S DONE, BUFFER IT IN RDATA
031C BC08                 MOV     BITCNT,#RDCNT      ;RESET BIT COUNTER
031E FB                   MOV     A,RSTAT            ;GET READ STATUS
031F 4301                 ORL     A,#RDYFLG          ;SET DATA READY FLAG
0321 AB                   MOV     RSTAT,A            ;RESTORE STATUS
0322 65          RDI4:    STOP    TCNT               ;STOP COUNTER
0323 23FA                 MOV     A,#RDTIM           ;GET TIMER CONSTANT (3/4 CELL TIME)
0325 62                   MOV     T,A                ;LOAD TIMER
0326 B933                 MOV     IRGCNT,#SLWIRG     ;LOAD IRG COUNT (READ USES SLOW SPEED)
0328 09          RDI7:    IN      A,P1               ;READ DRIVE STATUS
0329 9254                 JB4     RDI9               ;TEST IF CLEAR LEADER FOUND - ERROR
032B 564E                 JT1     RDI5               ;TEST INPUT LOOKING FOR EDGE
032D B650                 JFO     RDI6               ;INPUT=0, TEST LAST
032F E928        RDI8:    DJNZ    IRGCNT,RDI7        ;INPUT/LAST SAME, DEC IRG COUNT
0331 8902                 ORL     P1,#STP            ;COUNT EXPIRED, AT IRG, STOP DRIVE
0333 8A02                 ORL     P2,#DAOFF          ;TURN OFF DRIVE ACTIVE LED
0335 FB                   MOV     A,RSTAT            ;GET READ STATUS
0336 4308                 ORL     A,#IRGFLG          ;SET IRG FOUND FLAG
0338 AB                   MOV     RSTAT,A            ;RESTORE STATUS
                 ;
                 ;INTERRUPT EXIT ROUTINE - UPDATES FO IN STACK TO PRESERVE IT OVER RETR
                 ;
0339 C7          INTEXT:  MOV     A,PSW              ;GET CURRENT PSW FOR STACK POINTER
033A 5307                 ANL     A,#07H             ;LOOK AT STACK POINTER ONLY
033C 07                   DEC     A                  ;TRYING TO GET PSW ON TOP OF STACK
033D E7                   RL      A                  ;2 BYTES PER STACK ENTRY
033E 17                   INC     A                  ;POINT AT PSW ENTRY
033F 0308                 ADD     A,#08H             ;ADD OFFSET FOR POINTER
0341 A9                   MOV     IRGCNT,A           ;LOAD POINTER - USE IRGCNT REGISTER
0342 F1                   MOV     A,@IRGCNT          ;GET PSW
0343 B649                 JFO     EXIT1              ;TEST FO TO SEE WHAT TO SET FO TO
0345 53DF                 ANL     A,#0DFH            ;FO=0 THEREFORE RESET IT
0347 644B                 JMP     EXIT2
0349 4320        EXIT1:   ORL     A,#20H             ;FO=1 THEREFORE SET IT
034B A1          EXIT2:   MOV     @IRGCNT,A          ;RESTORE STACK
034C FF                   MOV     A,ASAVE            ;RECOVER A
034D 93                   RETR                       ;RETURN WITH RESTORE
                 ;
034E B62F        RDI5:    JFO     RDI8               ;INPUT=1, TEST LAST, SAME
0350 95          RDI6:    CPL     FO                 ;FINALLY DIFFERENT, SET FO TO CURRENT INPUT
0351 55                   STRT    T                  ;START TIMER
0352 6439                 JMP     INTEXT             ;EXIT
                 ;
0354 FB          RDI9:    MOV     A,RSTAT            ;GET READ STATUS
0355 4320                 ORL     A,#EOTFLG          ;SET EOT FLAG
0357 AB                   MOV     RSTAT,A            ;RESTORE STATUS
0358 8902                 ORL     P1,#STP            ;EOT SO STOP DRIVE
035A 8A02                 ORL     P2,#DAOFF          ;TURN OFF DRIVE ACTIVE LED
035C 6439                 JMP     INTEXT             ;EXIT
                 ;
                 $EJECT
```

```
ISIS-II MCS-48/UPI-41 MACRO ASSEMBLER, V3.0
DIGITAL CASSETTE CONTROLLER REV 1.0 - 26 MARCH 80

 LOC  OBJ         LINE        SOURCE STATEMENT

                  1025 ;**********************************************************************
                  1026 ;
                  1027 ;WRITE INTERRUPT ROUTINE
                  1028 ;
                  1029 ;**********************************************************************
                  1030 ;
 035E 23FC        1031 WRINT:   MOV     A,#WRTIM         ;GET WRITE TIME CONSTANT (1/2 CELL TIME)
 0360 62          1032          MOV     T,A              ;LOAD TIMER (IT'S STILL RUNNING)
 0361 B672        1033          JF0     WRINT1           ;TEST IF SECOND INT - DO NEXT BIT IF IT IS
 0363 0A          1034          IN      A,P2             ;FIRST INT - COMPLEMENT DATA OUT
 0364 126A        1035          JB0     WRI1
 0366 8A01        1036          ORL     P2,#DOHI
 0368 646C        1037          JMP     WRI2
 036A 9AFE        1038 WRI1:    ANL     P2,#DOLOW
 036C 95          1039 WRI2:    CPL     F0               ;SET SECOND INT FLAG
 036D 09          1040          IN      A,P1             ;TEST FOR CLEAR LEADER
 036E 92DC        1041          JB4     CLRLED           ;HANDLE IT IF IT'S FOUND
 0370 6439        1042          JMP     INTEXT           ;GO EXIT
                  1043 ;
                  1044 ;SECOND INTERRUPT FOR THIS BIT - GO GET NEXT BIT
                  1045 ;
 0372 FB          1046 WRINT1:  MOV     A,WSTAT          ;GET WRITE STATUS
 0373 52BA        1047          JB2     WRD4             ;TEST WRITE DONE FLAG - DONE IF YES
 0375 FD          1048          MOV     A,SERIAL         ;GET DATA REMAINDER
 0376 67          1049          RRC     A                ;ROTATE NEXT BIT INTO CARRY
 0377 AD          1050          MOV     SERIAL,A         ;RESTORE DATA
 0378 E683        1051          JNC     WRI3             ;TEST NEXT BIT
 037A 0A          1052          IN      A,P2             ;IF 0 - NO CHANGE IN OUTPUT
 037B 1281        1053          JB0     WRI4             ;IF 1 - COMPLEMENT OUTPUT
 037D 8A01        1054          ORL     P2,#DOHI
 037F 6483        1055          JMP     WRI3
 0381 9AFE        1056 WRI4:    ANL     P2,#DOLOW
 0383 85          1057 WRI3:    CLR     F0               ;RESET SECOND INT FLAG
 0384 EC39        1058          DJNZ    BITCNT,INTEXT    ;EXIT IF CHR NOT DONE
 0386 BC08        1059          MOV     BITCNT,#WRCNT    ;CHR IS DONE SO RESET BIT COUNTER
 0388 FB          1060          MOV     A,WSTAT          ;GET WRITE STATUS
 0389 12AA        1061          JB0     WRD2             ;TEST CHECKSUM FLAG
 038B 32B3        1062          JB1     WRD3             ;TEST SYNC FLAG
 038D C5          1063          SEL     RB0
                  1064 ;
                  1065 ;NO SPECIAL CHR BEING DONE - DEC BYTE COUNTER AND TEST IF DONE
                  1066 ;
 038E E894        1067          DJNZ    CNTLSB,WRI5      ;DEC LSB - IF NON-ZERO GET NEXT BYTE
 0390 F9          1068          MOV     A,CNTMSB         ;IF ZERO, GET MSB AND TEST IT
 0391 C6A1        1069          JZ      WRD1             ;IF MSB IS ZERO - DONE, GO WRITE CHECKSUM
 0393 C9          1070          DEC     CNTMSB           ;MSB IS NON-ZERO SO DEC IT
 0394 D6CA        1071 WRI5:    JNIBF   WRURUN           ;NOT DONE WITH ALL BYTES,
                  1072                                   ;IS THE NEW DATA IN DBBIN YET? NO - UNDERRUN
 0396 22          1073          IN      A,DBB            ;YES, READ IT
 0397 AC          1074          MOV     TEMP1,A          ;SAVE IT
 0398 76CF        1075          JF1     WCOMD            ;BE SURE IT WASN'T A COMMAND
 039A 6B          1076          ADD     A,CHKSUM         ;IT'S DATA, ADD TO CHECKSUM
 039B AB          1077          MOV     CHKSUM,A         ;RESTORE CHECKSUM
 039C FC          1078          MOV     A,TEMP1          ;GET DATA AGAIN
 039D D5          1079          SEL     RB1
 039E AD          1080          MOV     SERIAL,A         ;PUT DATA IN SERIALIZER
 039F 6439        1081          JMP     INTEXT           ;GO EXIT
                  1082 ;
                  1083 ;BYTE COUNT DONE - WRITE CHECKSUM
                  1084 ;
 03A1 FB          1085 WRD1:    MOV     A,CHKSUM         ;GET CHECKSUM
 03A2 D5          1086          SEL     RB1
 03A3 AD          1087          MOV     SERIAL,A         ;PUT IT IN SERIALIZER
 03A4 FB          1088          MOV     A,WSTAT          ;GET WRITE STATUS
 03A5 4301        1089          ORL     A,#CKSFLG        ;SET CHECKSUM FLAG
 03A7 AB          1090          MOV     WSTAT,A          ;RESTORE STATUS
 03A8 6439        1091          JMP     INTEXT           ;GO EXIT
                  1092 ;
                  1093 ;CHECKSUM BYTE DONE - DO SYNC
                  1094 ;
 03AA 53FE        1095 WRD2:    ANL     A,#NOT CKSFLG    ;RESET CHECKSUM FLAG
 03AC 4302        1096          ORL     A,#SYNFLG        ;SET SYNC FLAG
 03AE AB          1097          MOV     WSTAT,A          ;RESTORE STATUS
 03AF BDAA        1098          MOV     SERIAL,#SYNC     ;PUT SYNC IN SERIALIZER
 03B1 6439        1099          JMP     INTEXT           ;GO EXIT
                  1100 ;
                  1101 ;SYNC DONE - SET WRITE DONE FLAG
                  1102 ;
 03B3 53FD        1103 WRD3:    ANL     A,#NOT SYNFLG    ;RESET SYNC FLAG
 03B5 4304        1104          ORL     A,#WRDFLG        ;SET WRITE DONE FLAG
 03B7 AB          1105          MOV     WSTAT,A          ;RESTORE WRITE STATUS
 03B8 6439        1106          JMP     INTEXT           ;GO EXIT
                  1107 ;
                  1108 ;WRITE DONE FLAG FOUND SET - COMPLETELY DONE SO STOP AND RESET BUSY
                  1109 ;
 03BA BA00        1110 WRD4:    MOV     RESULT,#GOOD     ;GOOD RESULT
 03BC 8A01        1111 WRDON:   ORL     P2,#DOHI         ;SET OUTPUT TO 1
 03BE 65          1112          STOP    TCNT             ;STOP TIMER
 03BF 8902        1113          ORL     P1,#STP          ;STOP DRIVE
 03C1 8A02        1114          ORL     P2,#DAOFF        ;TURN OFF DRIVE ACTIVE LED
 03C3 D5          1115          SEL     RB1
 03C4 FB          1116          MOV     A,WSTAT          ;GET WRITE STATUS
 03C5 537D        1117          ANL     A,#NOT (WRFLG OR SYNFLG)         ;RESET WR/RD FLAG
 03C7 AB          1118          MOV     WSTAT,A          ;RESTORE STATUS
```

```
03C8 6439                  JMP     INTEXT           ;GO EXIT
                      ;
                      ;UNDERRUN OCCURRED
                      ;
03CA D5               WRURUN: SEL     RB1
03CB BA81                  MOV     RESULT,#UNDERW   ;UNDERRUN ERROR CODE
03CD 64BC                  JMP     WRDON            ;EXIT
                      ;
                      ;COMMAND FOUND WHEN DATA EXPECTED - CHECK IF ABORT
                      ;
03CF D5               WCOMD:  SEL     RB1
03D0 D305                  XRL     A,#ABORT         ;COMPARE TO ABORT
03D2 C6D8                  JZ      WC1              ;YES, THEN ABORT
03D4 BA82                  MOV     RESULT,#WCMDER   ;NO, DATA ERROR RESULT CODE
03D6 64BC                  JMP     WRDON            ;EXIT
03D8 BA01             WC1:    MOV     RESULT,#ABTCMP   ;ABORT COMPLETE RESULT COE
03DA 64BC                  JMP     WRDON
                      ;
                      ;CLEAR LEADER FOUNMD DURING WRITE
                      ;
03DC BA83             CLRLED: MOV     RESULT,#EOTERR   ;CLEAR LEADER ERROR CODE
03DE FB                    MOV     A,WSTAT          ;GET WRITE STATUS
03DF 4320                  ORL     A,#EOTFLG        ;SET EOT FLAG
03E1 AB                    MOV     WSTAT,A          ;RESTORE STATUS
03E2 64BC                  JMP     WRDON            ;EXIT
                      ;
                      END

USER SYMBOLS
ABORT  0005   ABTCMP 0001   ABTST  01DA   ABTST1 01DE   ARTJMP 0044   ASAVE  0007   B1     0017   B2     001F
BADCHS 0044   BEGIN  0009   BITCNT 0004   BLKCNT 0004   BLKSAV 0006   BLKTIM 0005   BOTFLG 0040   BUF1   01C7
BUF2   01CB   BUFFER 01C0   BUMPIT 02CC   BUSY   0080   CHKSUM 0003   CKSER  01BE   CKSFLG 0001   CLRLED 03DC
CLRTF  0059   CMDERR 0002   CMDIN  0023   CMDIN1 002C   CMDIN2 003A   CMDIN3 0033   CMDINJ 0292   CMDJMP 003E
CMDSAV 0002   CNTLSB 0000   CNTMSB 0001   DAOFF  0002   DAON   00FD   DEL1   00BA   DEL150 00C1   DEL2   00BC
DEL50  00B8   DESERL 0005   DOHI   0001   DOLOW  00FE   DR1    00CD   DRACT  0010   DRACTS 02BC   DRIVER 00C5
DRIVJ  014F   DUMPIT 02D3   EOTERR 0083   EOTFLG 0020   EXIT1  0349   EXIT2  034B   FASIRG 0020   FAST   00FB
FILPRT 0020   FORWD  00FE   GETDAT 0151   GOOD   0000   INT    0300   INTEXT 0339   IRGCNT 0001   IRGF1  01BA
IRGFLG 0008   IRGFND 0191   LBIN   0000   LBOUT  0000   LBRDY  0001   LBTST  0177   NDRACT 02C4   NINMSB 006D
NOFULL 0162   NT     00D0   NTAPE  0003   NWR    0004   OVERUN 0041   OVFLOW 02D0   PAS1   02DF   PAS2   02E3
PASHOL 02DC   RCLRTF 0103   RCMDER 0045   RD     00F7   RD1    0127   RD1A   012B   RD2    012D   RD3    0130
RD4    0135   RDATA  0006   RDCMD  0001   RDCNT  0008   RDERR  00D4   RDERR2 00F1   RDERR3 00DF   RDERR4 00E3
RDERR5 00E7   RDERR6 00F8   RDI1   030C   RDI2   0312   RDI3   0316   RDI4   0322   RDI5   034E   RDI6   0350
RDI7   0328   RDI8   032F   RDI9   0354   RDIBF  01D5   RDINT  0305   RDTIM  FFFA   RDYFLG 0001   READ   0100
REDJMP 0046   REOT   014B   REOTER 0046   RESCMD 0000   RESCOM 004E   RESET  0000   RESJMP 0044   RESULT 0002
REWIND 0001   REWJMP 004C   REWND  0294   REWND1 02A1   REWND2 02A8   REWND3 02AF   REWND4 02B8   RSKIP  0262
RSNB   018B   RSTAT  0003   RWCMD  0004   RWDIRG 0020   SERIAL 0005   SHIFIN 0316   SKCMD  0003   SKIP   0214
SKIP1  022B   SKIP11 0287   SKIP12 024F   SKIP13 025E   SKIP14 0257   SKIP2  022D   SKIP3  022F   SKIP4  023E
SKIP6  0244   SKIP7  024B   SKIP8  0278   SKIP9  0281   SKIPER 0200   SKIPR1 0206   SKIPR2 020C   SKIPR3 0212
SKPBOT 0048   SKPEOT 0047   SKPJMP 004A   SLOW   0004   SLWIRG 0033   SNBFLG 0002   SNBTST 0182   SRT    00FD
STAT   0007   STP    0002   STRFLG 0004   STSUP  00AD   SYNC   00AA   SYNC1  0042   SYNC2  0043   SYNFLG 0002
TAPIN  0040   TEMP0  0006   TEMP1  0004   TIMINT 0007   UNDERW 0081   UNFLOW 02D9   WC1    03D8   WCMDER 0082
WCOMD  03CF   WEOTER 00A9   WR     0008   WR1    005B   WR2    0061   WR3    008D   WR4    0098   WR5    009F
WRCMD  0002   WRCNT  0008   WRD1   03A1   WRD2   03AA   WRD3   03B3   WRD4   03BA   WRDFLG 0004   WRDON  03BC
WRFLG  0080   WRI1   036A   WRI2   036C   WRI3   0383   WRI4   0381   WRI5   0394   WRINT  035E   WRINT1 0372
WRITE  0055   WRTIM  FFFC   WRTJMP 0048   WRURUN 03CA   WSTAT  0003

ASSEMBLY COMPLETE,    NO ERRORS
```